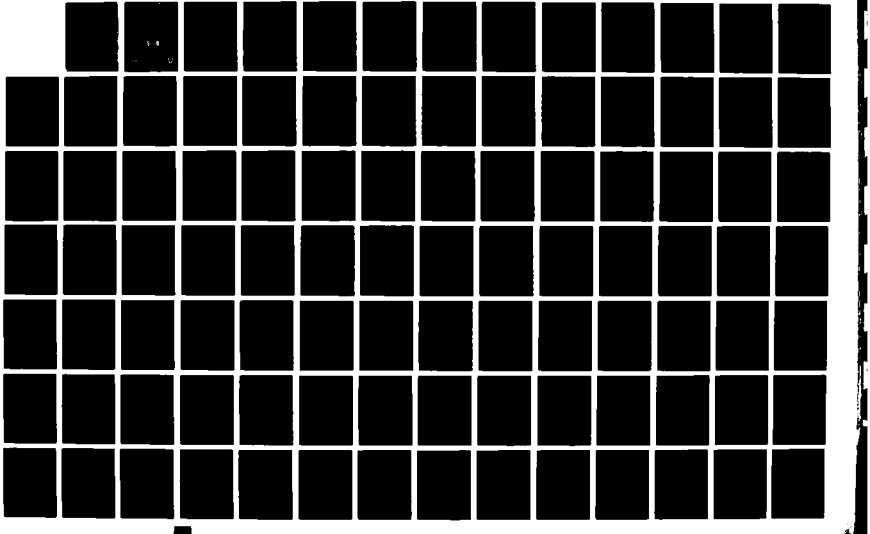
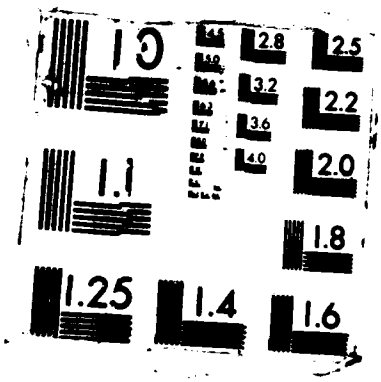
AD-A183 901 AN EXPONENTIAL FINITE DIFFERENCE TECHNIQUE FOR SOLVING PARTIAL DIFFERENTI.. (U) GAERTNER (W W) RESEARCH INC NORWALK CT R F HANDSCHUH JUN 87 NASA-E-3544 1/2

UNCLASSIFIED USAAVSCOM-TR-87-C-19 F/G 12/2 NL




2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.8
1.25
1.4
1.6

NASA
Technical Memorandum 89874

AVSCOM
Technical Report 87-C-19

An Exponential Finite Difference Technique for Solving Partial Differential Equations

Robert F. Handschuh
Propulsion Directorate
U.S. Army Aviation Research and Technology Activity—AVSCOM
Lewis Research Center
Cleveland, Ohio

June 1987

NASA

AN EXPONENTIAL FINITE DIFFERENCE TECHNIQUE FOR SOLVING PARTIAL DIFFERENTIAL EQUATIONS

Robert F. Handschuh

Propulsion Directorate
U.S. Army Aviation Research and Technology Activity - AVSCOM
Lewis Research Center
Cleveland, Ohio 44135

ABSTRACT

An exponential finite difference algorithm, as first presented by Bhattacharya for one-dimensional unsteady state, heat conduction in Cartesian coordinates, has been extended. The finite difference algorithm developed was used to solve the diffusion equation in one-dimensional cylindrical coordinates and applied to two- and three-dimensional problems in Cartesian coordinates. The method was also used to solve nonlinear partial differential equations in one (Burger's equation) and two (Boundary Layer equations) dimensional Cartesian coordinates. Predicted results were compared to exact solutions where available, or to results obtained by other numerical methods. It was found that the exponential finite difference method produced results that were more accurate than those obtained by other numerical methods, especially during the initial transient portion of the solution. Other applications made using the exponential finite difference technique included unsteady one-dimensional heat transfer with temperature varying thermal conductivity and the development of the temperature field in a laminar Couette flow.

ACKNOWLEDGEMENTS

I would like to thank Dr. Theo G. Keith, Jr. of the Department of Mechanical Engineering at the University of Toledo for his guidance and assistance throughout this research effort. My thanks also go to the members of the thesis review committee, Dr. Kenneth J. De Witt and Dr. Douglas Oliver.

TABLE OF CONTENTS

NOMENCLATURE

a_i, b_i, c_i	Thomas algorithm variables
B	Biot number
C_p	material specific heat, J/kg • K (Btu/lb_m • °F)
h	convection heat transfer coefficient, W/M^2 • °C (Btu/ft^2 • hr • °F)
i,j,k	nodal location in x,y, and z spatial coordinate directions respectively
J_0, J_1	Bessel functions of zero and first order
k	thermal conductivity, W/M • °C (Btu/hr • °F • ft)
k_i^n	thermal conductivity at i^{th} position, n^{th} time step, W/M • °C (Btu/hr • °F • ft)
L	distance between plates, M (ft)
M	dimensionless drive number
m	number of sub-intervals
N	number of nodes in a spatial direction
n	time step position designation
q	heat flux, W/M^2 (Btu/hr • ft^2)
r	spatial coordinate; cylindrical coordinates, M (ft)
T	temperature, °C (°F)
t	time, sec
Δt	time between time steps n and n + 1
U	Couette flow velocity, M/s (ft/s)
x,y,z	spatial coordinates, Cartesian coordinates, M (ft)
$\Delta x, \Delta y, \Delta z$	distance between nodal positions in the x,y, and z spatial directions
α	thermal diffusivity, M^2/s (ft^2/s)
β	rate of thermal conductivity variation
γ	$\Delta t/\rho C_p(\Delta x)^2$; $(W/M \cdot °C)^{-1}$ $((Btu/hr \cdot °F \cdot ft^2)^{-1})$

κ	constant
κ_1	constant used in exponential finite difference method with temperature varying thermal conductivity
δ_x	finite difference operator
λ_m	m^{th} eigenvalue of Bessel function
λ_i^n, ν_i^n	Thomas algorithm variables dependent on time step and spatial location
ξ	amplification factor
ν	kinematic viscosity, M^2/s (ft^2/s)
ρ	material density, kg/M^3 (lbm/ft^3)
φ, ψ, θ	separation variables
Ω	$\frac{\alpha \Delta t}{(\Delta x)^2}$ dimensionless time

I. INTRODUCTION

Partial differential equations have many important applications in the fields of engineering and physics. Many exact solutions exist depending on, the partial differential equation, the boundary conditions, and the number of spatial dimensions under consideration [1]. Coordinate systems other than Cartesian, more than one spatial dimension, and the boundary conditions all can pose problems that are either extremely difficult or impossible, to solve by analytical methods. Numerical methods thusly become the only possible solution method if the problem complexity is not to be compromised. Typically the ability of a particular method to predict a field variable is tested by numerically solving a problem for which a known exact solution is available. The ability of the method to predict the exact results is a measure of the confidence that can placed in a solution where no exact solution exists or experimental test results are unavailable.

The objective of the work to be presented is to extend, expand, and compare an explicit exponential finite difference technique first proposed by Bhattacharya [2]. To date the method has only been used for one-dimensional unsteady-state, heat transfer problems in Cartesian coordinates.

The method has been expanded, in this report, to allow application to a variety of problems. The exponential method will be extended here to the case of one-dimensional unsteady heat transfer in cylindrical

coordinates. Also, it was used in two- and three-dimensional unsteady heat condition. Other cases of interest that were solved using this approach include temperature varying conductivity in one-dimensional heat transfer and the development of the temperature field in laminar Couette flow. Solutions of the above cases were either compared to exact solutions or to results obtained by alternative numerical techniques.

One final application of the exponential finite difference algorithm was made for nonlinear partial differential equations. Burger's equation along with the boundary layer equations are solved using the exponential method. Thus, a demonstration of how to apply the method to nonlinear problems is described.

The results of all the different cases considered in this study indicated that the exponential technique produced results that were more accurate than those found through other numerical techniques. All exponential computer codes and those of the other competing numerical analysis were run in double precision on either the IBM-3033 or the Cray X-MP mainframes. The computer codes developed for the exponential finite difference method and other numerical techniques used for comparison are contained in the appendix of this report.

II. - ANALYSIS

The Exponential Finite Difference Algorithm

An explicit exponential finite difference algorithm as first derived by Bhattacharya [2] will now be presented. The method can be applied to many of the partial differential equations found in engineering and physics. The diffusion equation as it applies to conduction heat transfer will be used in the demonstration that follows. In reference [2-3] the method was derived for one-dimensional conduction heat transfer in Cartesian coordinates. To show how the method can be extended, a derivation parallel to the one presented in reference [2] will be made for unsteady state heat conduction in two-dimensions. Equations of this type are typically solved numerically by a variety of methods [4].

For two-dimensional heat transfer in Cartesian coordinates with constant material properties, the appropriate partial differential equation is [5]:

$$\frac{\partial T}{\partial t} = \alpha \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right) \tag{1}$$

To initiate the exponential method a product solution is assumed and is written as:

$$T(x,y,t) = \phi(x)\psi(y)\theta(t) \tag{2}$$

The initial conditions of the problem are assumed to be

$$\begin{aligned} T(x,y,0) &= f(x,y) \\ \theta(0) &= 1 \end{aligned} \tag{3}$$

Now taking the appropriate deviatives of Eq. (2) with respect to the independent variables,yields

$$\frac{\partial T}{\partial t} = \phi\psi \frac{\partial \theta}{dt} \; ; \quad \frac{\partial^2 T}{\partial x^2} = \psi\theta \frac{\partial^2 \phi}{\partial x^2} \; ; \quad \frac{\partial^2 T}{\partial y^2} = \phi\theta \frac{\partial^2 \psi}{\partial y^2} \tag{4}$$

Substituting Eq. (4) into Eq. (1) produces:

$$\phi\psi \frac{\partial \theta}{\partial t} = \alpha \left\{ \psi\theta \frac{\partial^2 \phi}{\partial x^2} + \phi\theta \frac{\partial^2 \psi}{\partial y^2} \right\}$$

Dividing both sides of the above by $\phi\psi\theta$ gives:

$$\frac{1}{\theta} \frac{\partial \theta}{\partial t} = \alpha \left\{ \frac{1}{\phi} \frac{\partial^2 \phi}{\partial x^2} + \frac{1}{\psi} \frac{\partial^2 \psi}{\partial y^2} \right\} = - \kappa \tag{5}$$

It can be seen that the variables have been separated. Consequently both sides of Eq. (5) must equal a constant, say, κ.

Now examining only the left hand side of Eq. (5),

$$\frac{1}{\theta} \frac{\partial \theta}{\partial t} = - \kappa$$

Multiplying the left-hand side of this equation by $\phi\psi/\phi\psi$ gives:

$$\frac{\phi\psi}{\theta\phi\psi} \frac{\partial \theta}{\partial t} = - \kappa$$

which can be rewritten from Eq. (4) as:

$$\frac{1}{T(x,y,t)} \frac{\partial T}{\partial t} = - \kappa$$

Direct integration produces:

$$T = c_2 \exp \{ - \kappa t \}$$

Next, the initial consition is used to evaluate the integration constant giving $c_2 = T(x,y,0)$; thus,

$$T(x,y,t) = T(x,y,0) \exp \{ - \kappa t \} \tag{6}$$

Returning to Eq. (5) only this time, the right-hand side is examined;

$$\alpha \left\{ \frac{1}{\phi} \frac{\partial^2 \phi}{\partial x^2} + \frac{1}{\psi} \frac{\partial^2 \psi}{\partial y^2} \right\} = - \kappa$$

Multiply this by $\frac{\theta\phi\psi}{\theta\phi\psi}$ and obtain

$$\alpha \left\{ \frac{\theta\psi}{\theta\phi\psi} \frac{\partial^2 \phi}{\partial x^2} + \frac{\theta\phi}{\theta\phi\psi} \frac{\partial^2 \psi}{\partial y^2} \right\} = - \kappa$$

or

$$\frac{\alpha}{\theta\phi\psi} \left\{ \theta\psi \frac{\partial^2 \phi}{\partial x^2} + \theta\phi \frac{\partial^2 \psi}{\partial y^2} \right\} = - \kappa$$

Equations (2) and (4) are now used which results in:

$$\frac{\alpha}{T} \left\{ \frac{\partial^2 T}{\partial x^2} + \frac{\partial T}{\partial y^2} \right\} = - \kappa \tag{7}$$

The temperature appearing as a coefficient is replaced using its initial value so that,

$$\frac{\alpha}{T(x,y,0)} \left\{ \frac{\partial^2 T}{\partial x^2} + \frac{\partial T}{\partial y^2} \right\} = - \kappa \tag{8}$$

The partial derivative terms can be written in central difference form about a node (i,j) as [6]:

$$\left. \begin{aligned} \frac{\partial^2 T}{\partial x^2} &\simeq \frac{T^n_{i+1,j} + T^n_{i-1,j} - 2T^n_{i,j}}{(\Delta x)^2} \\ \frac{\partial^2 T}{\partial y^2} &\simeq \frac{T^n_{i,j+1} + T^n_{i,j-1} - 2T^n_{i,j}}{(\Delta y)^2} \end{aligned} \right\} \tag{9}$$

Thus Eq. (8) becomes:

$$\frac{\alpha}{T^n_{i,j}} \left\{ \frac{T^n_{i+1,j} + T^n_{i-1,j} - 2T^n_{i,j}}{(\Delta x)^2} + \frac{T^n_{i,j+1} + T^n_{i,j-1} - 2T^n_{i,j}}{(\Delta y)^2} \right\} = - \kappa \tag{10}$$

Now Eq. (10) is substituted to replace the constant κ in the exponential of Eq. (6). Making the appropriate substitutions results in:

$$T_{i,j}^{n+1} = T_{i,j}^{n} \exp\left\{\frac{\alpha\Delta t}{T_{i,j}^{n}}\left[\frac{T_{i+1,j}^{n} + T_{i-1,j}^{n} - 2T_{i,j}^{n}}{(\Delta x)^2} + \frac{T_{i,j+1}^{n} + T_{i,j-1}^{n} - 2T_{i,j}^{n}}{(\Delta y)^2}\right]\right\}$$

If the grid spacing is constant ($\Delta x = \Delta y$), then the above equation can be written as:

$$T_{i,j}^{n+1} = T_{i,j}^{n} \exp\left\{\frac{\alpha\Delta t}{(\Delta x)^2}\left[\frac{T_{i+1,j}^{n} + T_{i-1,j}^{n} + T_{i,j+1}^{n} + T_{i,j-1}^{n} - 4T_{i,j}^{n}}{T_{i,j}^{n}}\right]\right\} \quad (11)$$

Note the Δt that appears in the above is the time elapsed between time steps n and $n+1$. The temperatures on the right-hand side are the four nearest neighbors to the i,j^{th} node (see Fig. 1).

In keeping with the notation derived by Bhattacharya [2], the term

$$\Omega = \frac{\alpha\Delta t}{(\Delta x)^2} \quad (12)$$

is called the dimensionless time step and

$$M_{i,j}^{n} = \frac{T_{i+1,j}^{n} + T_{i-1,j}^{n} + T_{i,j+1}^{n} + T_{i,j-1}^{n} - 4T_{i,j}^{n}}{T_{i,j}^{n}} \quad (13)$$

is called the dimensionless drive number. Thus, Eq. (11) can be rewritten rather compactly as:

$$T_{i,j}^{n+1} = T_{i,j}^{n} \exp\left\{\Omega M_{i,j}^{n}\right\} \quad (14)$$

It was found [2] that an improvement in the solution at the $n+1$ time step at node (i,j) can be made if the time step is divided into

a number of equal length time sub-intervals. The method of time step sub-intervals can be described as follows. Let us assume that the time interval will be divided into three intervals including the one at the end of the time step (Fig. 2). Now returning to Eq. (14), and evaluating at the n+1/3 time step, results in:

$$T_{i,j}^{n+1/3} = T_{i,j}^{n} \exp\left\{\frac{\Omega}{3} M_{i,j}^{n}\right\} \tag{15}$$

Proceeding from the n+1/3 time step to the n+2/3 time step:

$$T_{i,j}^{n+2/3} = T_{i,j}^{n+1/3} \exp\left\{\frac{\Omega}{3} M_{i,j}^{n+1/3}\right\} \tag{16}$$

Finally, the temperature can be written at the n+1 time step:

$$T_{i,j}^{n+1} = T_{i,j}^{n+2/3} \exp\left\{\frac{\Omega}{3} M_{i,j}^{n+2/3}\right\} \tag{17}$$

Now substituting Eq. (15) into Eq. (16) and substitution of Eq. (16) into Eq. (17), produces:

$$T_{i,j}^{n+1} = T_{i,j} \exp\left\{\frac{\Omega}{3} M_{i,j}^{n}\right\} \exp\left\{\frac{\Omega}{3} M_{i,j}^{n+1/3}\right\} \exp\left\{\frac{\Omega}{3} M_{i,j}^{n+2/3}\right\} \tag{18}$$

or

$$T_{i,j}^{n+1} = T_{i,j}^{n} \exp\left\{\frac{\Omega}{3}\left[M_{i,j}^{n} + M_{i,j}^{n+1/3} + M_{i,j}^{n+2/3}\right]\right\} \tag{19}$$

where the M's are then evaluated at the sub-time intervals and then summed for calculation of "T" at the n+1 time step. Equation (19) can be written more generally as:

$$T_{i,j}^{n+1} = T_{i,j}^{n} \exp\left\{\frac{\Omega}{m+1} \sum_{p=0}^{m} M_{i,j}^{n+p/(m+1)}\right\} \tag{20}$$

In reference [2], it was shown that for heat transfer applications, the time step can be subdivided as follows:

$$m = \begin{cases} \frac{N}{2} - 1 \ldots & \text{heat transfer coefficient} \to \infty \\ \frac{N}{2} + 1 \ldots & \text{finite heat transfer coefficient with flanking nodes outside calculation domain.} \end{cases} \quad (21)$$

N = number of nodes in one of the coordinate directions.

The procedure necessary to determine the dimensionless drive numbers will now be described. Since the method is an explicit technique, all information is known from the previous time step or from the previous time sub-interval. The effort is the centered around calculation of the drive numbers at the requested number of time sub-intervals for each of the spatial positions (nodes).

The calculation procedure is shown in Fig. (3). Because the drive numbers are evaluated at sub-time intervals, the temperature (or any other field variable) must also be known at these sub-time intervals. Therefore, the temperature field is calculated at each sub-time interval, and in turn is used to calculate the next drive number. The drive numbers for each node are summed for all the sub-time interval steps and then used to predict the field variable at the next complete time step. This results in a computer storage requirement of 4(N), where N is the number of nodes.

Extension To One-Dimensional Cylindrical Coordinates

In the previous section, the exponential finite difference technique was extended to two-dimensions. Now the procedure will be considered in another coordinate system. In particular the method will next be applied to radial one-dimensional, unsteady heat transfer.

The governing unsteady diffusion equation for a material with constant properties is:

$$\rho C_p \frac{\partial T}{\partial t} = k \left[\frac{1}{r} \frac{\partial}{\partial r} \left(r \frac{\partial T}{\partial r} \right) \right]$$

$$\frac{\partial T}{\partial t} = \alpha \left[\frac{\partial^2 T}{\partial r^2} + \frac{1}{r} \frac{\partial T}{\partial r} \right] \tag{22}$$

Assume that the temperature can be written as the product:

$$T(r,t) = \phi(r)\theta(t) \tag{23}$$

The initial conditions are

$$T(r,0) = f(r) \; ; \quad \theta(0) = 1 \tag{24}$$

Now taking the appropriate derivatives of Eq. (23); results in

$$\frac{\partial T}{\partial t} = \phi \frac{\partial \theta}{\partial t} \; ; \quad \frac{\partial T}{\partial r} = \theta \frac{\partial \phi}{\partial r} \; ; \quad \frac{\partial^2 T}{\partial r^2} = \theta \frac{\partial^2 \phi}{\partial r^2} \tag{25}$$

Substituting Eq. (25) into Eq. (23) yields:

$$\phi \frac{\partial \theta}{\partial t} = \alpha \left[\theta \frac{\partial^2 \phi}{\partial r^2} + \frac{1}{r} \theta \frac{\partial \phi}{\partial r} \right]$$

Dividing both sides by $\phi\theta$ brings:

$$\frac{1}{\theta} \frac{\partial \theta}{\partial t} = \alpha \left[\frac{1}{\phi} \frac{\partial^2 \phi}{\partial r^2} + \frac{1}{r\phi} \frac{\partial \phi}{\partial r} \right] = - \kappa \tag{26}$$

It can be seen that the variables have been separated. Thus, both sides of Eq. (26) must equal a constant, κ. Now using the time dependent side of the above , i.e.,

$$\frac{1}{\theta} \frac{\partial \theta}{\partial t} = - \kappa$$

and multiplying this by $\theta\phi/\theta\phi$ produces:

$$\frac{1}{T} \frac{\partial T}{\partial t} = - \kappa$$

Integrating for a particular value of the radial position "r" gives:

$$T = C_1 \exp \{ - \kappa t \}$$

Next the initial condition is used and the equation can be written as:

$$T(r,t) = T(r,0) \exp \{- \kappa t\} \tag{27}$$

Returning to Eq. (26) and using the radial side of the equation, we have

$$\alpha \left[\frac{1}{\phi} \frac{\partial^2 \phi}{\partial r^2} + \frac{1}{r\phi} \frac{\partial \phi}{\partial r} \right] = - \kappa$$

Multiplying by $\theta\phi/\theta\phi$, the equation can be rewritten as:

$$\alpha \left[\frac{1}{T} \frac{\partial^2 T}{\partial r^2} + \frac{1}{Tr} \frac{\partial T}{\partial r} \right] = - \kappa \tag{28}$$

Incorporating the initial condition,

$$\frac{\alpha}{T(r,0)} \left[\frac{\partial^2 T}{\partial r^2} + \frac{1}{r} \frac{\partial T}{\partial r} \right] = - \kappa \tag{29}$$

Next the partial derivatives are replaced using central finite differences [4]:

$$\frac{\partial^2 T}{\partial r^2} = \frac{T^n_{i+1} + T^n_{i-1} - 2T^n_i}{(\Delta r)^2}$$

$$\frac{\partial T}{\partial r} = \frac{T^n_{i+1} - T^n_{i-1}}{2\,\Delta r} \tag{30}$$

Substituting Eq. (30) into Eq. (29):

$$\frac{\alpha}{T^n_i} \left[\left(\frac{T^n_{i+1} + T^n_{i-1} - 2T^n_i}{(\Delta r)^2} \right) + \frac{1}{r_i} \left(\frac{T^n_{i+1} - T^n_{i-1}}{2\,\Delta r} \right) \right] = - \kappa \tag{31}$$

Equation (31) is used to replace the constant, $- \kappa$ in Eq. (27), i.e.,

$$T^{n+1}_i = T^n_i \exp \left\{ \frac{\alpha\,\Delta t}{T^n_i} \left[\left(\frac{T^n_{i+1} + T^n_{i-1} - 2T^n_i}{(\Delta r)^2} \right) + \frac{1}{r_i} \left(\frac{T^n_{i+1} - T^n_{i-1}}{2\,\Delta r} \right) \right] \right\}$$

Rearranging this brings:

$$T_i^{n+1} = T_i^n \exp\left\{\frac{\alpha\,\Delta t}{(\Delta r)^2}\left[\left(\frac{T_{i+1}^n + T_{i-1}^n - 2T_i^n}{T_i^n}\right) + \frac{\Delta r}{2r_i}\left(\frac{T_{i+1}^n - T_{i-1}^n}{T_i^n}\right)\right]\right\} \quad (32)$$

Equation (32) states that the temperature at the i^{th} radial position at the (n+1) time step is found from the product of temperature at i^{th} position, n^{th} time step and the exponential term that is composed of a dimensionless time step:

$$\Omega = \frac{\alpha\,\Delta t}{(\Delta r)^2} \quad (33)$$

and a dimensionless drive number:

$$M_i^n = \left(\frac{T_{i+1}^n + T_{i-1}^n - 2T_i^n}{T_i^n}\right) + \frac{\Delta r}{2r_i}\left(\frac{T_{i+1}^n - T_{i-1}^n}{T_i^n}\right) \quad (34)$$

It should be noted that this drive number applies to the interior nodes ($r_i \neq 0$). For the node at $r = 0$ the dimensionless drive number becomes:

$$M_i^n = \frac{2\left(T_{i-1}^n - T_i^n\right)}{T_i^n} \quad (35)$$

Finally Eq. (32) can be written in a compact form as

$$T_i^{n+1} = T_i^n \exp\left\{\Omega M_i^n\right\} \quad (36)$$

The sub-time interval evaluation for Eq. (36) is the same as that found in the two-dimensional Cartesian form as shown earlier.

Stability of the Exponential Finite Difference Method

With few exceptions, explicit finite difference procedures for solving partial differential equations are inherently unstable, unless certain numerical conditons are satisfied. These conditions take the

form of a grid size and/or time step requirement written in terms of parameters of the given problem. If these stability conditions are not met, the solution will diverge, often rather drastically. On the other hand, the stability requirements can make explicit methods impractical for a particular application by requiring an unrealistically small grid or time step. Nonetheless, these conditions must be known prior to the use of any explicit finite difference procedure.

There are a variety of methods that have been used to establish the stability constraints of a finite difference procedure: some are very elementary, some quite involved. In essence, the methods seek to find an expression for the amplification factor which is the ratio of the current solution result to that in the previous step. If the absolute value of the ratio is less than one the method is stable. Determination of the amplification factor for the exponential finite difference method is particularly convenient, as has been shown in [2]. For the one-dimensional cylindrical coordinate case, the amplification factor ξ can be readily defined as

$$\xi = \frac{T_i^{n+1}}{T_i^n} = \exp\left\{\frac{\alpha\,\Delta t}{(\Delta r)^2}\left[\left(\frac{T_{i+1}^n + T_{i-1}^n - 2T_i^n}{T_i^n}\right) + \frac{\Delta r}{2r_i}\left(\frac{T_{i+1}^n - T_{i-1}^n}{T_i^n}\right)\right]\right\} \tag{37}$$

or from Eq. (32) for an interior node:

$$\xi = \frac{T_i^{n+1}}{T_i^n} = \exp\left\{\Omega M_i^n\right\} \tag{38}$$

So for stability to exist as Δt and Δr approach zero:

$$\lim_{\substack{\Delta t \to 0 \\ \Delta r \to 0}} |\xi| \leq 1 \tag{39}$$

To satisfy this requirement the exponent in the eponential of Eq. (38) must obviously be less than or equal to zero. Since the components that make up Ω in that exponent are all positive, this implies that the dimensionless drive number will dictate whether or not the stability criteria is met. For the cylindrical coordinate case the dimensionless drive number must satisfy:

$$M_i^n = \left(\frac{T_{i+1}^n + T_{i-1}^n - 2T_i^n}{T_i^n} \right) + \frac{\Delta r}{2r_i} \left(\frac{T_{i+1}^n - T_{i-1}^n}{T_i^n} \right) \leq 0 \tag{40}$$

Multiplying by T_i^n this becomes:

$$\left(T_{i+1}^n + T_{i-1}^n - 2T_i^n \right) + \frac{\Delta r}{2r_i} \left(T_{i+1}^n - T_{i-1}^n \right) \leq 0$$

Define $\beta \equiv \Delta r/2r_i$ and rearrange to get

$$\begin{aligned} (1 + \beta) T_{i+1}^n + (1 - \beta) T_{i-1}^n &\leq 2T_i^n \\ T_i^n &\geq \frac{1}{2} \left[(1 + \beta) T_{i+1}^n + (1 - \beta) T_{i-1}^n \right] \end{aligned} \tag{41}$$

Equation (41) needs to be satisfied otherwise an unstable condition can exist. In Eq. (41) as $\Delta r \to 0$, or equivalently $\beta \to 0$, the stability condition becomes:

$$T_i^n \geq \frac{1}{2} \left[T_{i+1}^n + T_{i-1}^n \right] \tag{42}$$

Also for one-dimensional cylindrical coordinate case, the node at $r = 0$ has a different stability criteria because the node at $i+1$ does not exist. Since the radial derivative of the field variable must equal zero at $r = 0$. In finite difference terms this can be realized

by requiring that $T^n_{i+1} = T^n_{i-1}$ at this node. Thus, at the origin, Eq. (42) becomes

$$T^n_1 \geq T^n_{i-1} \tag{43}$$

As stated by Bhattacharya [2], the dimensionless drive number is the determining factor whether or not the stability criteria is met. However, the dimensionless time, if made large enough, could cause the solution to become unstable. Since time sub-interval division is used, the total dimensionless time step Ω could become quite large. From [2] it was recommended that the dimensionless time step be chosen to satisfy the following:

$$\frac{\Omega}{m+1} \leq 0.5 \tag{44}$$

where m is the number of time step sub-intervals involved in the calculations. If $\Omega = 5$, for example, then m would have to be greater than or equal to 9 i.e., nine sub-intervals would be required. From Eq. (21) with infinite heat transfer coefficent, a minimum of 20 nodes would be needed.

Another useful comparison to be made is the one-node model as used in Refs. [2] and [7] where the value of the dependent variable at the surrounding nodes is set equal to zero. For the one-node model, as stated in Ref. [7], the exponential finite difference and exact are the same (Fig. 4). This figure indicates that the exponential solution remains stable as Ω is increased.

Effect of Initial and Boundary Conditions on the Exponential Finite Difference Method

In this section the effect of boundary conditions on the exponential finite difference technique will be investigated. Boundary conditions that are typical of heat transfer applications will be considered. The conditions to be presented are: (1) finite heat transfer coefficient (mixed condition), (2) infinite heat transfer coefficient (Dirichlet condition), (3) constant heat flux (Neumann condition), and (4) time varying. Initial conditions, where the field variable is equal to zero, will also be discussed.

Finite Heat Transfer Coefficient

For this boundary condition, the method used in reference [2] will be utilized. Numerical implementation of the boundary condition requires that a node be placed outside the solid in the surrounding medium. This external node will be used in the finite difference equation at the solid surface. One-dimensional heat transfer in a cylinder ($T = T(r,t)$) will be used to demonstrate the procedure.

Using the exponential finite difference method for $T = (r,t)$, it was found earlier (Eq. (36)) that

$$T_1^{n+1} = T_1^n \exp\left\{\Omega M_1^n\right\} \tag{45}$$

where the dimensionless time is given by

$$\Omega = \frac{\alpha \, \Delta t}{(\Delta r)^2}$$

and the dimensionless drive number by:

$$M_i^n = \left\{ \left(\frac{T_{i+1}^n + T_{i-1}^n - 2T_i^n}{T_i^n} \right) + \frac{\Delta r}{2r_i} \left(\frac{T_{i-1} - T_{i+1}}{T_i^n} \right) \right\} \qquad (46)$$

The thermal condition at the surface is found by equating the conductive and convective heat fluxes at the surface:

$$- k \left. \frac{\partial T}{\partial r} \right|_{r=R} = h \left(T \Big|_{r=R} - T_\infty \right) \qquad (47)$$

Using the node numbering as shown in Fig. 5 and a central difference to express the deverative, Eq. (47) becomes:

$$h \left(T_1^n - T_\infty \right) = - k \left(\frac{T_0^n - T_2^n}{2 \, \Delta r} \right) \qquad (48)$$

Solving for T_0^n, the temperature of the external node, yields

$$T_0^n = T_2^n + \frac{2h \, \Delta r}{k} \left(T_\infty - T_1^n \right) \qquad (49)$$

Let $B = h \, \Delta r/k$ (Biot number [8]), thus Eq. (49) is written:

$$T_0^n = T_2^n + 2BT_\infty^n - 2BT_1^n \qquad (50)$$

Now that an expression for the temperature at the external node is known, it can be used in the expression for the temperature at the surface node. This results in:

$$T_1^{n+1} = T_1^n \exp \left\{ \Omega \, M_1^n \right\}$$

where (51)

$$M_1^n = \left\{ \left[\frac{T_2^n - 2T_1^n + 2BT_\infty + T_2^n - 2BT_1^n}{T_1^n} \right] + \frac{\Delta r}{R} \left[\frac{B \left(T_\infty^n - T_1^n \right)}{T_1^n} \right] \right\}$$

and the drive number can be further simplified to:

$$M_1^n = \left\{\left[\frac{2\,T_2^n - (1 + B)T_1^n + BT_\infty}{T_1^n}\right] + \frac{\Delta r B}{R}\left[\frac{T_\infty - T_1^n}{T_1^n}\right]\right\} \qquad (52)$$

Infinite heat transfer coefficient

If the heat transfer coefficient in Eq. (47) is placed on the left-hand side of the equation and then allowed to become infinite, it is seen that the surface temperature will equal the temperature of the surroundings. Thus, the boundary condition in which $h \to \infty$ is identical to that in which a boundary temperature is held constant. In the calculation procedure, these isothermal boundary nodes are only needed for calculation of the temperature field at the surrounding nodes. For example in a two-dimensional square grid with a total of 121 nodes calculation would be reduced to a total of 81 nodes if the temperature is specified for all four boundaries.

Constant Heat Flux

For a constant heat flux applied to the boundary surface, the same procedure as was used in the finite heat transfer coefficient case will be utilized. The condition at the surface for one-dimensional cylindrical coordinates is given by:

$$q = k \left.\frac{\partial T}{\partial r}\right|_{r=R} \qquad (53)$$

An external node is placed outside the solid as was done earlier. Equation (53) can then be written

$$q = k\left(\frac{T_0^n - T_2^n}{2\,\Delta r}\right) \qquad (54)$$

Solving for the external node temperature results in:

$$T_0^n = \frac{2q\ \Delta r}{k} + T_2^n \tag{55}$$

At the surface the exponential finite difference equation is:

$$T_1^{n+1} = T_1^n \exp\left\{\Omega\left[\left(\frac{T_0^n + T_2^n - 2T_1^n}{T_1^n}\right) + \frac{\Delta r}{2R}\left(\frac{T_0^n - T_2^n}{T_1^n}\right)\right]\right\}$$

and substituting in Eq. (55):

$$T_1^{n+1} = T_1^n \exp\left\{\Omega\left[\left(\frac{\frac{2q\ \Delta r}{k} + T_2^n + T_2^n - 2T_1^n}{T_1^n}\right) + \frac{\Delta r}{2R}\left(\frac{\frac{2q\ \Delta r}{k} + T_2^n - T_2^n}{T_1^n}\right)\right]\right\}$$

Rearranging this produces:

$$T_1^{n+1} = T_1^n \exp\left\{\Omega\left[\frac{2\left(\frac{q\ \Delta r}{k} + T_2^n - T_1^n\right)}{T_1^n} + \frac{(\Delta r)^2\, q}{RkT_1^n}\right]\right\} \tag{56}$$

Time Varing Boundary Conditions

This condition is similar to the constant boundary temperature condition except that the boundary temperature must be incremented as the calculation marches in time. The boundary condition must reside in the time step loop which is shown as the outer most loop in Fig. 3. The temperatures on these boundaries are incremented and held constant as the subtime interval calculations are made.

Dependent Variable Initially Equal to Zero

One last condition that can exist wherever there is initial zero temperature ($T(x,0) = 0$) needs to be discussed. If this condition is encountered, then the following substitution should be made or else the exponential finite difference method will not work. This can be readily seen by examing any of the numerical equations e.g., Eq. (56). Since the initial temperature would appear in the denominator of the

exponent in the exponential, problems would ensue. To circumvent this difficulty define a new variable $\bar{T}$, such that $\bar{T}(x,t) = 1.0 - T(x,t)$. Now the exponential finite difference equations described above can be utilized by simple replacement of the T variable with the $\bar{T}$ variable.

III. NUMERICAL COMPARISON OF THE EXPONENTIAL FINITE DIFFERENCE METHOD TO EXACT SOLUTIONS AND OTHER NUMERICAL TECHNIQUES

The final product of any numerical study is how well the given method performs when compared to known exact solutions or to other numerical techniques. The exponential finite difference method will now be applied to the following cases to demonstrate the capability of the method to solve the diffusion equation:

(i) One-dimensional heat conduction in cylindrical coordinates with an infinite and a finite heat transfer coefficient at the surface, unsteady state,

(ii) Two-dimensional unsteady state heat conduction in Cartesian coordinates,

(iii) Solution of Laplaces equation, Cartesian coordinates,

(iv) One-dimensional heat conduction in Cartesian coordinates, unsteady state with temperature varying thermal conductivity,

(v) Steady state Couette flow,

(vi) Three-dimensional heat conduction in Cartesian coordinates, unsteady state.

One-Dimensional Heat Conduction in Cylindrical Coordinates

The one-dimensional cylindrical coordinate heat conduction case with temperature as a function of time and radial position will be investigated for infinite and finite heat transfer coefficient. The exact results for both cases can be found in Ref. [9].

For infinite heat transfer coefficient on the boundary surface the exact result is given in [9] as:

$$\frac{T(r,t) - T_\infty}{T_0 - T_\infty} = 2 \sum_{\bar{m}=1}^{\infty} \frac{e^{-\alpha\lambda_{\bar{m}}^2 t}\, J_0(\lambda_{\bar{m}} r)}{(\lambda_{\bar{m}} r)\, J_1(\lambda_{\bar{m}} R)} \tag{57}$$

where $\lambda_{\bar{m}} R$ is the $\bar{m}^{th}$ zero of

$$J_0(\lambda_{\bar{m}} R) = 0 \tag{58}$$

The results of both the exact analysis and the exponential finite difference method are shown in Table I. As can be seen from the tabulated results, exponential finite difference results approach the exact solution as the number of nodes is increased or as the dimensionless time step is decreased.

When the heat transfer coefficient has a finite value at the surface, the exact solution from [9] is:

$$\frac{T(r,t) - T_\infty}{T_0 - T_\infty} = 2B \sum_{\bar{m}=1}^{\infty} \frac{e^{-\alpha\lambda_{\bar{m}}^2 t}\, J_0(\lambda_{\bar{m}} r)}{\left(\lambda_{\bar{m}}^2 R^2 + B^2\right) J_0(\lambda_{\bar{m}} R)} \tag{59}$$

where $B = hR/k$ (Biot number) and $\lambda_{\bar{m}}$ (characteristics values) are given by (for cooling):

$$(\lambda_{\bar{m}} R) J_1(\lambda_{\bar{m}} R) - B J_0(\lambda_{\bar{m}} R) = 0 \tag{60}$$

The results are shown in Table II for various values of the Biot number. As would be expected the solution approaches the exact solution as the number of nodes in increased. The size of the Biot number did not seem to effect the accuracy of the solution. As the elapsed time of the solution proceeded, temperatures predicted by the

exponential finite difference method approached the exact result. Also the results indicated that reducing the size of the time sub-interval increased the method's accuracy.

One last comparison will be made while investigating the exponential finite difference technique in one-dimensional cylindrical coordinates. The problem situation is shown in Fig. (6) and applies to a cylindrical annulus with the following initial and boundary conditions:

$$\begin{aligned} T(r,0) &= 0 \\ T(R_2,t) &= 1.0 \\ \frac{\partial T}{\partial r}(R_1,t) &= 0 \end{aligned} \tag{61}$$

In Ref. [4] this problem was solved numerically using a characteristic-value solution. A comparison of results is shown in Table III for the exponential method using the same grid spacing as in [4] and for the case where grid spacing is halved. The results are seen to compare quite well with the finer mesh being slightly closer to the value from Ref. [4] especially during the first few time steps of the solution.

Two-Dimensional Heat Conduction in Cartesian Coordinates

The exponential finite difference technique will now be applied to the two-dimensional heat conduction problem in Cartesian coordinates shown in Fig. (7). The exponential finite difference method will be compared to the solution of this problem, as performed in Ref. [4], using the alternating direction implicit technique (ADI). The results of the two numerical techniques and the exact analysis are shown in Table IV. The temperature indicated for comparison is that at the

origin $x = y = 0$, shown in Fig. (7). As maybe seen, the ADI technique does not predict the temperature as accurately as the exponential finite difference method at the first time value shown in Table IV. However, as the amount of elapsed time increases either method does a very good job at predicting this temperature. When the number of grid points was increased, by halving the spatial intervals, the exponential finite difference method was found to be more accurate for all the time steps.

Since the ADI method is one that requires simultaneous solution of equations in the two coordinate directions, the time step size can be made large. The exponential finite difference technique must have the dimensionless time step kept below 0.25 to keep the solution stable. So the required CPU time for the exponential method is higher for this application.

Solution of Laplaces Equation

Since the exponential finite difference method has been used for two-dimensional unsteady state conduction, a natural extension with little additional effort would be to use this method to solve Laplace's equation. This can be implemented in the exponential finite difference method by just allowing the solution to march in time until no further change in the field variable is indicated.

As an example, the problem as shown in Fig. (8) will be solved and the results compared to those given in Ref. [4]. In the referenced work, the solution was found by using a Gauss-Seidel iterative technique.

A comparison of results along a diagonal from the position (x = 0, y = 1) to (x = 1, y = 0) is presented in Table V for two different grid spacings. As can be seen, the solutions are nearly identical with the exponential method requiring a smaller number of iterations (or time step increments) to reach a similiar result.

One-Dimensional Unsteady State Conduction With Temperature Varing Thermal Conductivity

The effect of temperature varing thermal conductivity will now be investigated using three different numerical schemes: a pure explicit, the exponential method and an implicit technique. The problem to be solved is illustrated in Fig. (9a). The thermal conductivity as shown in Fig. (9b) is assumed to be a linear function of temperature.

The exponential finite difference method will be applied first to the given problem. The governing partial differential equation is [1]:

$$\rho C_p \frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left(k \frac{\partial T}{\partial x}\right) \tag{62}$$

From [1], Eq. (62) can be changed to a simpler form by using a new variable θ (the Kirchoff transformation) given by:

$$\theta = \frac{1}{k_r}\int_{T_R}^{T} k(T)dT \tag{63}$$

where k_R is the conductivity at temeprature T_R, and

$$\frac{\partial \theta}{\partial t} = \frac{k}{k_R}\frac{\partial T}{\partial t} \quad \text{or} \quad \frac{\partial T}{\partial t} = \frac{k_R}{k}\frac{\partial \theta}{\partial t}$$
$$\frac{\partial \theta}{\partial x} = \frac{k}{k_R}\frac{\partial T}{\partial x} \tag{64}$$

Substituting Eq. (64) into (62) gives:

$$\frac{\rho C_p k_R}{k}\left(\frac{\partial\theta}{\partial t}\right) = \frac{\partial}{\partial x}\left(k_R \frac{\partial\theta}{\partial x}\right)$$

or,

$$\frac{\rho C_p}{k}\left(\frac{\partial\theta}{\partial t}\right) = \frac{\partial^2\theta}{\partial x^2} \tag{65}$$

Since it has been assumed that the thermal conductivity is a linear function of temperature,

$$k(T) = k_R(1 + \beta T) \tag{66}$$

Now returning to Eq. (63) and substituting in Eq. (66), we have:

$$\theta = \frac{1}{k_R}\int_{T_R}^{T} (k_R + \beta k_R T)dT$$

Direct integration yields:

$$\theta = (T - T_R)\left\{1 + \frac{\beta}{2}(T + T_R)\right\} \tag{67}$$

Equation (67) provides the relationship between the variable T and the new variable θ.

Returning to Eq. (65),

$$\frac{\partial\theta}{\partial t} = \frac{k}{\rho C_p}\frac{\partial^2\theta}{\partial x^2} \tag{68}$$

Equation (68) is in a form now that the exponential finite difference can be applied. The resulting equation in the variable can be shown to be given by:

$$\theta_i^{n+1} = \theta_i^n \exp\left\{\frac{\Delta t}{\rho C_p(\Delta x)^2}\left[\frac{k_i^n\left(\theta_{i+1}^n + \theta_{i-1}^n - 2\theta_i^n\right)}{\theta_i^n}\right]\right\} \tag{69}$$

Evaluating Eq. (67) at node i brings

$$\theta_i^n = \left(T_i^n - T_R\right)\left\{1 + \frac{\beta}{2}\left(T_i^n + T_R\right)\right\}$$

or

$$\theta_i^n = \left(T_i^n - T_R\right) + \frac{\beta}{2}\left(\left(T_i^n\right)^2 + T_R^2\right) \tag{70}$$

Substitution of Eq. (70) into Eq. (69) at the appropriate time steps and nodal locations will give:

$$\left(T_i^{n+1} - T_R\right) + \frac{\beta}{2}\left(\left(T_i^{n+1}\right)^2 - T_R^2\right) = \left[\left(T_i^n - T_R\right) + \frac{\beta}{2}\left[\left(T_i^n\right)^2 - T_R^2\right]\right] \cdot$$

$$\exp\left\{\gamma k_i^n \left[\frac{\left(T_{i+1}^n + T_{i-1}^n - 2T_i^n\right) + \frac{\beta}{2}\left[\left(T_{i+1}^n\right)^2 + \left(T_{i-1}^n\right)^2 - 2\left(T_i^n\right)^2\right]}{\left(T_i^n - T_R\right) + \frac{\beta}{2}\left[\left(T_i^n\right)^2 - T_R^2\right]}\right]\right\} \tag{71}$$

where

$$\gamma = \frac{\Delta t}{\rho C_p (\Delta x)^2}$$

If $T_R = T_\infty = 0.0$, Eq. (71) becomes:

$$T_i^{n+1} + \frac{\beta}{2}\left(T_i^{n+1}\right)^2 = \left(T_i^n + \frac{\beta}{2} T_i^n\right) \cdot$$

$$\exp\left\{\gamma k_i^n \left[\frac{\left(T_{i+1}^n + T_{i-1}^n\right) - 2T_i^n + \frac{\beta}{2}\left[\left(T_{i+1}^n\right)^2 + \left(T_{i-1}^n\right)^2 - 2\left(T_i^n\right)^2\right]}{T_i^n + \frac{\beta}{2}\left(T_i^n\right)^2}\right]\right\} \tag{72}$$

The equation for T_i^{n+1} is a quadratic with the right-hand side of the equation all being known at time step n, so define a variable κ_i such that

$$\kappa_i = \left[T_i^n + \frac{\beta}{2}\left(T_i^n\right)^2\right] \exp\left\{\gamma k_i^n \cdot \left[\frac{\left(T_{i+1}^n + T_{i-1}^n - 2T_i^n\right) + \frac{\beta}{2}\left[\left(T_{i+1}^n\right)^2 + \left(T_{i-1}^n\right)^2 - 2\left(T_i^n\right)^2\right]}{T_i^n + \frac{\beta}{2}\left(T_i^n\right)^2}\right]\right\} \tag{73}$$

Equation (72) then becomes:

$$\left(T_1^{n+1}\right)^2 + \frac{2}{\beta} T_1^{n+1} - \frac{2}{\beta} \kappa_1 = 0 \tag{74}$$

Solving this and using the positive root results in:

$$T_1^{n+1} = \frac{1}{\beta}\left(-1 + \sqrt{1 + 2\kappa_1\beta}\right) \tag{75}$$

where

$$\beta > 0$$

Equation (75) in conjunction with Eq. (74) are implemented in the exponential finite difference solution sequence. In this case the conductivity as well as the temperature field must be kept track of on the sub-time interval level. The dimensionless time step, Ω, and the rate of conductivity change, β, must be both considered when choosing the step size so the solution does not become unstable. For this method, the term $(\gamma k_1^n/m+1)$ in the exponential was considered at its maximum possible value and the time step was adjusted to retain stability. This criteria was chosen so that

$$\frac{\gamma k_1^n}{m+1} < 0.5$$

The next method to be investigated for the temperature varying conductivity problem will be the pure explicit method. As stated earlier the governing partial differential equation for one-dimensional conduction is given by:

$$\rho C_p \frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left(k \frac{\partial T}{\partial x}\right) \tag{76}$$

Using the chain rule this equation can be put into a nonconservative form.

$$\rho C_p \frac{\partial T}{\partial t} = k \frac{\partial^2 T}{\partial x^2} + \frac{\partial k}{\partial T} \left(\frac{\partial T}{\partial x}\right)^2 \tag{77}$$

Assuming the same linear profile as before:

$$\left.\begin{aligned} k(T) &= k_R + \beta k_R T \\ \text{then} \\ \frac{\partial k}{\partial T} &= \beta k_R \end{aligned}\right\} \tag{78}$$

Substituting Eq. (78) into (77) will give:

$$\rho C_p \frac{\partial T}{\partial t} = k_R(1 + \beta T) \frac{\partial^2 T}{\partial x^2} + \beta k_R \left(\frac{\partial T}{\partial x}\right)^2$$

or

$$\frac{\partial T}{\partial t} = \alpha_R(1 + \beta T) \frac{\partial^2 T}{\partial x^2} + \beta \alpha_R \left(\frac{\partial T}{\partial x}\right)^2 \tag{78}$$

where

$$\alpha_R = \frac{k_R}{\rho C_p}$$

Using central space differences and a forward time difference we may write:

$$\left.\begin{aligned} \frac{\partial T}{\partial x} &= \frac{T_{i+1}^n - T_{i-1}^n}{2\Delta x} \\ \frac{\partial T}{\partial t} &= \frac{T_i^{n+1} - T_i^n}{\Delta t} \\ \frac{\partial^2 T}{\partial x^2} &= \frac{T_{i+1}^n + T_{i-1}^n - 2T_i^n}{(\Delta x)^2} \end{aligned}\right\} \tag{79}$$

Substituting Eq. (79) into (78) produces:

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} = \alpha_R\left(1 + \beta T_i^n\right)\left[\frac{T_{i+1}^n + T_{i-1}^n - 2T_i^n}{(\Delta x)^2}\right] + \beta \alpha_R \left[\frac{T_{i+1}^n - T_{i-1}^n}{2\Delta x}\right]^2$$

This may be simplified and written as:

$$T_i^{n+1} = T_i^n + \Omega \left[\left(1+\beta T_i^n\right) \left(T_{i+1}^n + T_{i-1}^n - 2T_i^n\right) + \frac{\beta}{4} \left(T_{i+1}^n - T_{i-1}^n\right)^2 \right] \tag{80}$$

where

$$\Omega = \frac{\alpha_R \, \Delta t}{(\Delta x)^2}$$

Equation (80) can be used to directly solve for the temperature field at the next time step. Some additional care must be used to keep the solution stable as the size of the dimensionless time step Ω, and the rate of conductivity change, β both effect the solution.

The final method to be implemented for comparative purposes is the implicit method. Starting with Eq. (78), we have:

$$\frac{\partial T}{\partial t} = \alpha_R \, (1 + \beta T) \, \frac{\partial^2 T}{\partial x^2} + \beta \alpha_R \left(\frac{\partial T}{\partial x}\right)^2 \tag{81}$$

To avoid a solution sequence that would require the solution of nonlinear algerbraic equations, the following will be assumed;

(a) The term $(1 + \beta T)$ can be replaced with $(1 + \beta T_i^n)$

(b) The squared first derivative term can be replaced by

$$\left(\frac{\partial T}{\partial x}\right)^2 = \left(\frac{T_{i+1}^n - T_{i-1}^n}{2\Delta x}\right) \left(\frac{T_{i+1}^{n+1} - T_{i-1}^{n+1}}{2\Delta x}\right)$$

This is a linearizing technique known as lagging the coefficients. Substituting these into Eq. (81) will produce:

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} = \alpha_R \left(1 + \beta T_i^n\right) \left[\frac{T_{i+1}^{n+1} + T_{i-1}^{n+1} - 2T_i^{n+1}}{(\Delta x)^2} \right] + \alpha_R \beta \left(\frac{T_{i+1}^n - T_{i-1}^n}{2\Delta x}\right) \left(\frac{T_{i+1}^{n+1} - T_{i-1}^{n+1}}{2\Delta x}\right)$$

Further simplification gives:

$$T_i^{n+1} - T_i^n = \Omega\left(1 + \beta T_i^n\right)\left(T_{i+1}^{n+1} + T_{i-1}^{n+1} - 2T_i^{n+1}\right) + \frac{\Omega\beta}{4}\left(T_{i+1}^n - T_{i-1}^n\right)\left(T_{i+1}^{n+1} - T_{i-1}^{n+1}\right) \qquad (82)$$

where

$$\Omega = \frac{\alpha\,\Delta t}{(\Delta x)^2}$$

Now define:

$$\begin{aligned} \lambda_i^n &= \frac{\beta}{4}\left(T_{i+1}^n - T_{i-1}^n\right) \\ \eta_i^n &= 1 + \beta T_i^n \end{aligned} \qquad (83)$$

Substituting Eq. (83) into (82) yields:

$$T_i^{n+1} - T_i^n = \Omega\left(\eta_i^n\left\{T_{i+1}^{n+1} + T_{i-1}^{n+1} - 2T_i^{n+1}\right\} + \lambda_i^n\left\{T_{i+1}^{n+1} - T_{i-1}^{n+1}\right\}\right)$$

Simplifying this results in:

$$T_i^{n+1}\left(1 + 2\Omega\eta_i^n\right) - T_{i+1}^{n+1}\left(\Omega\eta_i^n + \Omega\lambda_i^n\right) - T_{i-1}^{n+1}\left(\Omega\eta_i^n - \Omega\lambda_i^n\right) = T_i^n \qquad (84)$$

The equation shown above is now in a form that can be used in the Thomas Algorithm [4], i.e., Eq. (84), can be written as:

$$a_i T_{i-1}^{n+1} + b_i T_i^{n+1} + c_i T_{i+1}^{n+1} = T_i^n$$

where

$$\left.\begin{aligned} a_i &= -\Omega\left(\eta_i^n - \lambda_i^n\right) \\ b_i &= 1 + 2\Omega\eta_i^n \\ c_i &= -\Omega\left(\eta_i^n + \lambda_i^n\right) \end{aligned}\right\} \qquad (85)$$

Equation (85) can now be solved using a tri-diagonal matrix routine. The variables a_i, b_i, and c_i must be evaluated at each position and time step as their values change as the field variable T changes.

A comparison of results of the three methods can be found in Fig. (10) and Table VI. Figure (10) shows the temperature field through the slab cross-section. From this, it is evident that the exponential and pure explicit methods give very similiar results. The implicit method predicted higher temperatures closer to the slab surface and lower temperature at the slab centerline then either of the two explicit methods. In Table VI the results at the slab center are shown for various elapsed times. As can be seen, all three methods agreed with each other to within a few percent.

The Steady State Temperature Field of a Couette Flow

Another application of the exponential finite difference method will now be presented. The problem to be investigated is the developing temperature field in laminar Couette flow [7]. The problem statement is illustrated in Fig. (11). Neglecting viscous effects, the governing equation is given by [10]:

$$\rho C_p U_x \frac{\partial T}{\partial x} = k\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right) \tag{86}$$

Neglecting conduction in the x-direction or assuming that the convection term is much greater than the conduction term, Eq. (86) becomes:

$$U_x \frac{\partial T}{\partial x} = \alpha \frac{\partial^2 T}{\partial y^2}$$
$$\alpha = \frac{k}{\rho C_p} \tag{87}$$

From Fig. (11) using the expression for the velocity in the x-direction, $U_x = Uy/L$, Eq. (87) becomes

$$\frac{\partial T}{\partial x} = \frac{\alpha L}{Uy} \frac{\partial^2 T}{\partial y^2} \tag{88}$$

Equation (88) is now in a form where separation of variables can be implemented. Following the same procedure as indicated earlier to find the exponential finite difference equation, it can be shown that:

$$T_j^{i+1} = T_j^i \exp\left\{ \frac{\Delta x \alpha L}{U(\Delta y)^2 y_j} \left(\frac{T_{j+1}^i + T_{j-1}^i - 2T_j^i}{T_j^i} \right) \right\} \tag{89}$$

The procedure utilized here is that the solution marches in the x-direction instead of time as was the case for the previous examples. Information from the last x-position and y-direction are used to determine the dependent variable at the next x-position.

Results of implementing this method are shown in Fig. (12). The temperature field is shown for three x-locations for two different values of L/U. The results indicate that as the upper plate velocity U is increased, the propogation of the temperature change in the y-direction is slowed down.

Unsteady State Heat Conduction in Three-Dimensional Coordinates

The final application of the exponential finite difference method to the diffusion equation will be that of three-dimensional, unsteady state heat conduction. The exponential method, a pure explicit method, and an implicit method (method of Douglas, [11]) will be compared to exact solution for the situation shown in Fig. (13).

The exact solution to the problem illustrated in Fig. 13 is given in Ref. [10] as:

$$\frac{T(x,y,z,t) - T_1}{T_0 - T_1} = 8 \sum_{m=0}^{\infty} \sum_{n=0}^{\infty} \sum_{p=0}^{\infty} \left(\frac{(-1)^{m+n+p}}{\left(m + \frac{1}{2}\right)\left(n + \frac{1}{2}\right)\left(p + \frac{1}{2}\right)} \right) \cdot$$

$$\cdot \exp \left\{ - \left(\frac{\left(m + \frac{1}{2}\right)^2}{a^2} + \frac{\left(n + \frac{1}{2}\right)^2}{b^2} + \frac{\left(p + \frac{1}{2}\right)^2}{c^2} \right) \pi^2 \alpha t \right\}$$

$$\cdot \cos \left[\left(m + \frac{1}{2}\right) \frac{\pi x}{a} \right] \cos \left[\left(n + \frac{1}{2}\right) \frac{\pi y}{b} \right] \cos \left[\left(p + \frac{1}{2}\right) \frac{\pi z}{c} \right] \qquad (90)$$

where a, b, and c are the widths of the cube in the x, y, and z directions respectively. Equation (90) will be used to determine how well the numerical techniques predict the temperature distribution.

The exponential finite difference technique will be investigated first. The sequence to be followed for determining the finite difference equation is the same as presented for the earlier cases. The procedure for this three-dimensional case consists of the following stepped procedure:

(1) Linearize the partial differential equation

(2) Assume a product solution

(3) Separate time from spatial dependence

(4) Solve for time dependence

(5) Insert the appropriate spatial finite differences into exponential term that results from step 3

Based on this procedure the three-dimensional exponential finite difference equation can be shown to be:

$$T^{n+1}_{i,j,k} = T^{n}_{i,j,k} \exp\left\{\Omega\left[\left(\frac{T^{n}_{i+1,j,k} + T^{n}_{i-1,j,k} - 2T^{n}_{i,j,k}}{T^{n}_{i,j,k}}\right) + \left(\frac{T^{n}_{i,j+1,k} + T^{n}_{i,j-1,k} - 2T^{n}_{i,j,k}}{T^{n}_{i,j,k}}\right) + \left(\frac{T^{n}_{i,j,k+1} + T^{n}_{i,j,k-1} - 2T^{n}_{i,j,k}}{T^{n}_{i,j,k}}\right)\right]\right\} \tag{91}$$

Using the sub-time interval concept, Eq. (91) becomes:

$$T^{n+1}_{i,j,k} = T^{n}_{i,j,k} \exp\left\{\frac{\Omega}{m+1}\sum_{p=0}^{m} M^{n+p/(m+1)}_{i,j,k}\right\} \tag{92}$$

where m is the number of subtime intervals, Ω is the dimensionless time step, and $M^{n}_{i,j,k}$ is the dimensionless drive number given by.

$$M^{n}_{i,j,k} = \left(\frac{T^{n}_{i+1,j,k} + T^{n}_{i-1,j,k} - 2T^{n}_{i,j,k}}{T^{n}_{i,j,k}}\right) + \left(\frac{T^{n}_{i,j+1,k} + T^{n}_{i,j-1,k} - 2T^{n}_{i,j,k}}{T^{n}_{i,j,k}}\right) + \left(\frac{T^{n}_{i,j,k+1} + T^{n}_{i,j,k-1} - 2T^{n}_{i,j,k}}{T^{n}_{i,j,k}}\right) \tag{93}$$

Equation (92) will be used for all interior nodes in Fig. 13. This equation, as well as those that result from the other analysis, will be adjusted along the insulated boundaries to take into account the boundary condition that exists there.

The next method to be applied to this three-dimensional case will be the pure explicit method. The finite difference equation for this method is given by [11]:

$$T^{n+1}_{i,j,k} = T^{n}_{i,j,k}(1 - 6\Omega) + \Omega\left(T^{n}_{i+1,j,k} + T^{n}_{i-1,j,k} + T^{n}_{i,j+1,k} + T^{n}_{i,j-1,k} + T^{n}_{i,j,k+1} + T^{n}_{i,j,k-1}\right) \tag{94}$$

where $\Omega = \frac{\alpha \Delta t}{(\Delta x)^2}$; $\Delta x = \Delta y = \Delta z$

As shown in Ref. [11] the dimensionless time step Ω must be:

$$\Omega \leq \frac{1}{6} \tag{95}$$

to ensure stability of the method.

The last numerical technique that will be applied is the Method of Douglas [11]. This method is implicit, and the spatial directions are considered sequentially in the x, y, and then z directions respectively. The intermediate temperatures U (found from x-direct sweep) and V (found from y-direction sweep) are used to calculate the actual temperature field variable T (found from z-direction sweep). The equations that are solved sequentially are presented as follows.

$$\frac{U_{i,j,k} - T^n_{i,j,k}}{\alpha \Delta t} = \frac{1}{2} \delta_x^2 \left(U_{i,j,k} + T^n_{i,j,k} \right) + \delta_y \left(T^n_{i,j,k} \right) + \delta_z \left(T^n_{i,j,k} \right) \tag{96}$$

$$\frac{V_{i,j,k} - T^n_{i,j,k}}{\alpha \Delta t} = \frac{1}{2} \delta_x^2 \left(U_{i,j,k} + T^n_{i,j,k} \right) + \frac{1}{2} \delta_y^2 \left(V_{i,j,k} + T^n_{i,j,k} \right) + \delta_z^2 \left(T^n_{i,j,k} \right) \tag{97}$$

$$\frac{T^{n+1}_{i,j,k} - T^n_{i,j,k}}{\alpha \Delta t} = \frac{1}{2} \delta_x^2 \left(U_{i,j,k} + T^n_{i,j,k} \right) + \frac{1}{2} \delta_y^2 \left(V_{i,j,k} + T^n_{i,j,k} \right) + \frac{1}{2} \delta_z^2 \left(T^{n+1}_{i,j,k} + T^n_{i,j,k} \right) \tag{98}$$

where the finite difference operator in the x-direction , for example, would be:

$$\delta_x^2 = \frac{T^n_{i+1,j,k} + T^n_{i-1,j,k} - 2T^n_{i,j,k}}{(\Delta x)^2} \tag{99}$$

Equations (96), (97), and (98) must be solved successively because the variable U is used in equation (97) to find V and so on. Since the

method operates on one spatial direction at a time, the Thomas Algorithm can be utilized. In the case of finding the U variable, the y and z nodal positions are held constant for all the x-direction nodal positions (Fig. 14). This process is repeated until all y and z nodal values for the x-direction variable U are calculated. This procedure is then repeated in a similar way for the V variable and then finally for the actual temperature field variable.

The results from the three different, three-dimensional solution methods are shown in Table VII. The exponential finite difference method described above outperformed the pure explicit and the method of Douglas for all positions as shown in Table VII.

In Ref. [11] nine different methods to solve the diffusion equation in three dimensions were investigated. The method of Douglas was the preferred method because of its accurate results and low computer CPU time. In that study the pure explicit method required the lowest amount of CPU time with the method of Douglas requiring approximately four times as much. In the present study all three methods were run on two different mainframe computers to investigate how well these three methods compared in CPU times. The results are shown in Table VIII. All three methods were exercised for the same number of time steps. As indicated, the exponential method was approximately three times faster than the method of Douglas but still slower than pure explicit method. From these results it could be concluded that the exponential method would have been chosen as the preferred method had it been used in competition with the nine numerical methods as described in Ref. [11].

IV. - EXTENSION OF THE EXPONENTIAL FINITE DIFFERENCE METHOD TO NONLINEAR PARTIAL DIFFERENTIAL EQUATIONS

The exponential finite difference method will now be applied to two different nonlinear problems. The problems to be addressed will be the viscous Burger's equation and the boundary layer equations (steady state flow over a flat plate).

The viscous Burger's equation is given in Ref. [12] as:

$$\frac{\partial U}{\partial t} + U \frac{\partial U}{\partial x} = \nu \frac{\partial^2 U}{\partial x^2} \tag{100}$$

To allow application of the exponential method to this equation, the equation must be first linearized. So letting $U = A$ = constant, for the nonlinear, term and rearranging the equation, gives:

$$\frac{\partial U}{\partial t} = - A \frac{\partial U}{\partial x} + \nu \frac{\partial^2 U}{\partial x^2} \tag{101}$$

Assuming a product solution of the form

$$U(x,t) = \phi(x)\theta(t)$$

and taking the appropriate derivatives, Eq. (101) becomes

$$\phi \frac{\partial \theta}{\partial t} = - A\theta \frac{\partial \phi}{\partial x} + \nu\theta \frac{\partial^2 \phi}{\partial x^2}$$

Division by $\phi\theta$ gives:

$$\frac{1}{\theta} \frac{\partial \theta}{\partial t} = - \frac{A}{\phi} \frac{\partial \phi}{\partial x} + \frac{\nu}{\phi} \frac{\partial^2 \phi}{\partial x^2} = - \kappa = \text{constant} \tag{102}$$

It can be seen that the terms are now separated. As has been shown earlier, the left-hand side of Eq. (102) can be written as:

$$U(x,t) = U(x,0) \exp\{-\kappa t\} \tag{103}$$

Now returning to Eq. (102) and examining the x-dependence, we have

$$-\frac{A}{\phi}\frac{\partial \phi}{\partial x} + \frac{\upsilon}{\phi}\frac{\partial^2 \phi}{\partial x^2} = -\kappa$$

Multiplying both sides by $\phi\theta$ gives:

$$-A\theta\frac{\partial \phi}{\partial x} + \upsilon\theta\frac{\partial^2 \phi}{\partial x^2} = -\kappa\theta\phi$$

This can be written in finite difference form as:

$$\frac{1}{U_i^n}\left[-U_i^n\left(\frac{U_{i+1}^n - U_{i-1}^n}{2\,\Delta x}\right) + \upsilon\left(\frac{U_{i+1}^n + U_{i-1}^n - 2U_i^n}{(\Delta x)^2}\right)\right] = -\kappa \tag{104}$$

This is used to replace the exponent in Eq. (103), thus

$$U_i^{n+1} = U_i^n \exp\left\{\frac{\Delta t \upsilon}{(\Delta x)^2}\left[-\frac{\Delta x}{2\upsilon}\left(U_{i+1}^n - U_{i-1}^n\right) + \left(\frac{U_{i+1}^n + U_{i-1}^n - 2U_i^n}{U_i^n}\right)\right]\right\} \tag{105}$$

Equation (105) is the exponential finite difference equation for the viscous Burger's equation. An example will now be used to demonstrate the method.

An exact steady state solution to Burger's equation is available for the following conditions

$$U(0,t) = U_0$$

$$U(L,t) = 0$$

The steady-state solution was given as [12]:

$$U(x) = U_0 U_1 \left\{\frac{1 - \exp\left(\bar{U} R_e\left(\frac{x}{L} - 1\right)\right)}{1 + \exp\left(\bar{U} R_e\left(\frac{x}{L} - 1\right)\right)}\right\}$$

where

$$Re_L = \frac{U_0 L}{\nu} \tag{106}$$

and U_1 is the solution of the equation

$$\frac{U_1 - 1}{U_1 + 1} = \exp\left\{- U_1 \, Re_L\right\}$$

The exponential finite difference method will be now used to numerically solve the problem stated above. However, for the stated conditions, a problem arises with the portion of the velocity field initially at zero. To overcome this difficulty, the substitution method described earlier will be used. A new variable will be defined such that

$$\bar{U} = U_0 - U$$

and Burger's equation then becomes:

$$\frac{\partial \bar{U}}{\partial t} = (\bar{U} - U_0) \frac{\partial \bar{U}}{\partial x} + \nu \frac{\partial^2 \bar{U}}{\partial x^2} \tag{107}$$

with the following imposed conditions, if ($U_0 = 1$):

$$\bar{U}(0,t) = 0$$

$$\bar{U}(L,t) = U_0 \tag{108}$$

Using Eq. (107) the same method of separation of variables must be performed on the $\bar{U}$ variable. The problem is now solved for the $\bar{U}$ variable and the substitution shown above is then made to find the U variable. The exponential finite difference equation for $\bar{U}$ can be shown to be:

$$U_i^{n+1} = U_i^n \exp\left\{\frac{\Delta t \nu}{(\Delta x)^2}\left[\left(\frac{-\Delta x}{2\nu}\right)\left[\frac{\left(1 - U_i^n\right)\left(U_{i+1}^n - U_{i-1}^n\right)}{U_i^n}\right] + \frac{\left(U_{i+1}^n + U_{i-1}^n - 2U_i^n\right)}{U_i^n}\right]\right\} \qquad (109)$$

The results obtained by applying Eq. (109) and the conditions in Eq. (108) are compared to the steady state exact results of Eq. (106) and are shown in Fig. (15). The results from the exponential method were nearly the same as the exact method. The exponential method was allowed to march in time for quite a number of steps without special treatment of the dimensionless group $\Delta t\nu/(\Delta x)^2$ which could have been altered to allow convergence to the actual solution in less time steps.

Another application of Burger's equation was made to investigate the effect of the diffusion term. The results for the variation of ν over four orders of magnitude are shown in Fig. (16) for the same instant in time. At the two lower ν values, the total range of the field variable takes place over a small number of nodal positions. A better approximation could be made for these cases by using a finer grid. Also included on Fig. (16) is the solution of Burger's equation by a pure explicit technique. For the value of ν chosen, the solution oscillates around the predicted solution found from the exponential method. The pure explicit solution was found using the same number of nodes and the same dimensionless step size. When the solution oscillates, as the pure explicit solution did, the resulting velocity field can contain physically impossible values.

The last application to be investigated will be for the development of a laminar boundary layer on a flat plate (Fig. (17)). In Ref. [10] the steady state formulation is given in terms of the following three partial differential equations:

for continuity:

$$\frac{\partial U}{\partial x} + \frac{\partial V}{\partial y} = 0 \tag{110}$$

for momentum:

$$U \frac{\partial U}{\partial x} + V \frac{\partial U}{\partial y} = \nu \frac{\partial^2 U}{\partial y^2} \tag{111}$$

for energy:

$$U \frac{\partial T}{\partial x} + V \frac{\partial T}{\partial y} = \alpha \frac{\partial^2 T}{\partial y^2} \tag{112}$$

with the boundary conditions:

$$\begin{aligned} U(x,o) &= 0 \qquad U(o,y) = U_o \\ V(x,o) &= 0 \qquad V(x,L) = 0 \\ T(x,o) &= 0 \qquad T(o,y) = T_o \end{aligned} \tag{113}$$

ν and α are the momentum and thermal diffusivities respectively.

Equations (111) and (112) can be solved by using the method presented for the viscous Burger's equation. The only difference is that the solution will march in the x-direction instead of time. Keeping this procedure in mind, results of the separation of variables for Eqs. (111) and (112) were found to be:

$$U_j^{i+1} = U_j^i \exp\left\{\frac{\Delta x}{U_j^i}\left[-\frac{V_j^i}{U_j^i}\left(\frac{U_{j+1}^i - U_{j-1}^i}{2\,\Delta y}\right) + \frac{\nu}{U_j^i}\left(\frac{U_{j+1}^i + U_{j-1}^i - 2U_j^i}{(\Delta y)^2}\right)\right]\right\} \tag{114}$$

$$T_j^{i+1} = T_j^i \exp\left\{\frac{\Delta x}{T_j^i}\left[\frac{-V_j^i}{U_j^i}\left(\frac{T_{j+1}^i - T_{j-1}^i}{2\,\Delta y}\right) + \frac{\alpha}{U_j^i}\left(\frac{T_{j+1}^i + T_{j-1}^i - 2T_j^i}{(\Delta y)^2}\right)\right]\right\} \tag{115}$$

The continuity equation is written as [12]:

$$V_j^{i+1} = V_j^{i+1} - \frac{\Delta y}{2\,\Delta x}\left(U_j^{i+1} - U_j^i + U_{j-1}^{i+1} - U_{j-1}^i\right) \qquad (116)$$

Equations (114) and (115) are first solved using a spatial sub-increment as was done for the cases when time was the marching direction of the solution. After this, the continuity Eq. (116), is then solved.

The results of this application are shown in Fig. (18) for a Prandtl number equal to 0.72. As can be seen the thermal boundary layer was outside the velocity boundary layer as would be expected. The results with the Prandtl number equal to 0.72 were compared to the exact solution as presented in Ref. [10]. A downstream position was chosen and the results are compared in Table IX. The exponential method results were in good agreement with the exact results.

CONCLUDING REMARKS

In conclusion, an exponential finite difference technique has been extended to other coordinates systems and expanded to handle problems in two and three dimensions. The method has direct application to linear partial differential equations such as the diffusion equation and can be extended to solve nonlinear equations.

The method was applied to the following cases:

(1) One-dimensional, unsteady state heat transfer in cylindrical coordinates, infinite and finite heat transfer coefficient.

(2) Two- and three-dimensional, unsteady state heat transfer in Cartesian coordinates.

(3) One dimensional heat transfer, with temperature varying thermal conductivity.

(4) Developing temperature field in laminar Couette flow.

(5) Nonlinear partial differential equations (Burger's equation and boundary layer equations)

The exponential finite difference method predicted the field variable with a higher degree of accuracy in those cases examined where the exact solution was available. When extended to three dimensions, the accuracy was still higher for the exponential finite difference but the computer CPU time was increased. When the exponential method was compared to other numerical techniques, the results were found to be very compariable.

In conclusion, the results predicted for the exponential finite difference algorithm for the cases presented in this study demonstrated that:

(1) Field variable was predicted with a higher degree of accuracy than other numerical techniques where exact solutions exist.

(2) The method can be applied to linear and nonlinear partial differential equations with dependent variables that can be separated.

(3) The stability of the method is the same as that of pure explicit methods, where the sub-time interval step size determines the stability.

REFERENCES

[1] Carslaw, H., Jaeger, J.: Conduction of Heat in Solids, Oxford University Press, Ely House, London W., 1959.

[2] Bhattacharya, M.: "An Explicit Conditionally Stable Finite Difference Equation for Heat Conduction Problems," International Journal for Numerical Methods in Engineering, Vol. 21, pp. 239-265, 1985.

[3] Bhattacharya, M.: "A New Improved Finite Difference Equation for Heat Transfer During Transient Change," Applied Mathematical Modeling, Vol. 10, 1986.

[4] Carnahan, B., Luther, H., Wilkes, J.: Applied Numerical Methods, John Wiley & Sons, Inc., New York, NY, 1969.

[5] Hildebrand, F.: Advanced Calculus for Applications, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1976.

[6] Patankar, S.: Numerical Heat Transfer and Fluid Flow, McGraw-Hill Book Company, New York, NY, 1980.

[7] Patankar, S., Babiza, B.: "A New Finite-Difference Scheme for Parabolic Differential Equations," Numerical Heat Transfer, Vol. 1, pp. 27-37, 1978.

[8] Holman, J.: Heat Transfer, McGraw Hill Book Company, New York, NY, 1976.

[9] Arpaci, V.: Conduction Heat Transfer, Addison-Wesley Publishing Company, Menlo Park, CA, 1966.

[10] Bird, R., Stewart, W., Lightfoot, E.: Transport Phenomena, John Wiley & Sons, New York, NY, 1960.

[11] Thiabault, J.: "Comparison of Nine Three-Dimensional Numerical Methods for the Solution of the Heat Diffusion Equation," Numerical Heat Transfer, Vol. 8, pp. 281-298, 198_.

[12] Anderson, D., Tannehill, J., Pletcher, R., Computational Fluid Mechanics and Heat Transfer, McGraw-Hill Book Company, New York, NY, 1984.

TABLE I. - COMPARISON OF RESULTS FOR DIFFERENT DIMENSIONLESS TIME STEP FOR ONE-DIMENSIONAL HEAT TRANSFER IN CYLINDRICAL COORDINATES WITH INFINITE HEAT TRANSFER COEFFICIENT AT THE SURFACE. INITIAL AND BOUNDARY CONDITIONS ARE:

[$h \to \infty$, $T(r,o) = 1.0$, $T(R,t) = 0.0$, $\Omega = \alpha\, \Delta t/(\Delta r)^2$, $\alpha = 1.0$ M^2/s, N = number of nodes, m = number of sub-time intervals.]

t, sec	From surface r-distance (M)	N = 11 m = 4 $\Omega = 1.0$	N = 21 m = 9 $\Omega = 1.0$	N = 21 m = 9 $\Omega = 2.0$	N = 21 m = 9 $\Omega = 5.0$	Exact analysis ref. [9]
0.1	0.1	0.127004	0.126768	0.126819	--------	0.126669
.1	1.0	.862431	.852204	.853083	--------	.848368
.5	.1	.011959	.011671	.011680	.011715	.011582
.5	1.0	.094334	.090309	.090379	.090652	.088895
.5		Total 50 steps $\frac{\Omega}{M+1} = 0.2$	Total 200 steps $\frac{\Omega}{M+1} = 0.1$	Total 100 steps $\frac{\Omega}{M+1} = 0.2$	Total 40 steps $\frac{\Omega}{M+1} = 0.5$	

TABLE II. - FINITE HEAT TRANSFER COEFFICIENT CYLINDRICAL COORDINATES WITH THE FOLLOWING CONDITIONS: $T(r,o) = 1.0$, $T_\infty = 0$, $\Omega = \alpha\, \Delta t/(\Delta r)^2$

Time	h/k	R	Exact ref. [9]	Exponential finite difference results		
				N = 11 m = 4 $\Omega = 1.0$	N = 21 m = 9 $\Omega = 5.0$	N = 21 m = 9 $\Omega = 1.0$
0.1	1	1	0.6846	0.7073	0.6978	- ---
		0	.9768	.9814	.9797	------
.2	1	1	.5702	.5976	.5857	------
		0	.8702	.8852	.8780	-- ---
.4	1	1	.4132	.4441	.4303	------
		0	.6420	.6698	.6563	------
.1	2	1	.5009	.5285	.5199	- ----
		0	.9594	.9670	.9643	------
.1	5	1	.2558	.2777	-- ---	0.2669
		0	.9265	.9385	- ----	.9306

TABLE III. - COMPARISON OF EXPONENTIAL FINITE DIFFERENCE METHOD IN ONE-DIMENSIONAL CYLINDRICAL COORDINATES TO THE RESULTS OF REFERENCE [4].

[α = 1.0, Δt = 1.0 sec, $\alpha \Delta t/\Delta r^2$ = 1.0. N = number of nodes, m = number of sub-intervals

Time, sec	h/k	R, in.	Reference [4]	N = 10 m = 4 Ω = 1.0	N = 19 m = 8 Ω = 1.0
5	∞	18	0.77220	0.773094	0.772922
		10	.01449	.011353	.011951
10	∞	18	.84661	.846719	.846811
		10	.11595	.112112	.113523
30	∞	18	.93546	.935278	.935521
		10	.57722	.575979	.578198
90	∞	18	.99370	.993686	.993776
		10	.95872	.958596	.959245

TABLE IV. - COMPARISON OF EXPONENTIAL FINITE DIFFERENCE METHOD IN TWO-DIMENSIONAL CARTESIAN COORDINATES TO THE ALTERNATING DIRECTION IMPLICIT METHOD [4].

[For comparison to results in ref. [4] at x = y = o.]

Time, sec	Exact	ADI [4]	Ω = 1.0, N = 11, Δt = 0.01, m = 4	Ω = 1.0, N = 21, Δt = 0.0025, m = 9
0.1	0.09883	0.09333	0.09829	0.09924
.2	.40354	.40354	.40256	.40354
.3	.63179	.63224	.63080	.63166
.4	.77486	.77532	.77403	.77472
.5	.86252	.86283	.86187	.86240
.6	.91607	.91624	.91559	.91597
.7	.94877	.94886	.94841	.94869
.7			(Total of 70 steps)	(Total of 280 steps)
			$\frac{\Omega}{m+1} = 0.2$	$\frac{\Omega}{m+1} = 0.1$

TABLE V. - STEADY STATE HEAT TRANSFER IN TWO-DIMENSIONS

Comparison of exponential finite difference technique to a Gauss-Seidel technique for the solution of Laplace's equation.

x	y	9 by 9 Grid		5 by 5 Grid	
		Gauss-Seidel 88 iterations(a)	Exponential finite difference 40 iterations	Gauss-Seidel 22 iterations(a)	Exponential finite difference 20 iterations
0.000	1.000	0.0	0.0	0.0	0.0
.125	.875	1.7413	1.7414	--------	--------
.250	.750	6.8946	6.8949	7.1428	7.1430
.375	.625	15.0330	15.0335	--------	--------
.500	.500	24.9999	25.0004	25.0000	25.0003
.625	.375	34.9667	34.9672	--------	--------
.750	.250	43.1052	43.1055	42.8571	42.8573
.875	.125	48.2587	48.2588	--------	--------
1.000	0.000	100.000	100.000	100.000	100.000

[a]From reference [4].

TABLE VI. - COMPARISON OF EXPONENTIAL, PURE-EXPLICIT, AND IMPLICIT FINITE DIFFERENCE METHODS FOR ONE-DIMENSIONAL, UNSTEADY-STATE HEAT TRANSFER WITH TEMPERATURE VARYING THERMAL CONDUCTIVITY AT THE CENTER OF THE SLAB

[$K(T) = 1.0 + \beta(T)$; $\beta = 0.01$.]

Time, sec	Temperature, °C		
	Exponential finite difference, $N = 11$ $m = 4$, $\Omega = 0.5$, $\Delta t = 0.005$ sec	Pure explicit, $N = 11$, $\Omega = 0.25$, $\Delta t = 0.0025$ sec	Implicit, $\Omega = 1.0$ $\Delta t = 0.01$ sec
0.01	98.15998	100.00000	94.35768
.02	88.87177	89.21321	85.90591
.05	61.30161	60.09306	61.31385
.1	34.37147	33.41929	35.37178

TABLE VII. - COMPARISON OF THREE DIFFERENT, THREE-DIMENSIONAL UNSTEADY STATE HEAT TRANSFER SOLUTIONS

[$T(x,y,z,o) = 1.0$; $T(x,y,L,t) = T(x,L,z,t) = T(L,y,z,t) = 0$; $\partial T/\partial x(o,y,z,t) = \partial T/\partial y(x,o,z,t) = \partial T/\partial z(x,y,o,t) = 0$; N = number of nodes in x, y, and z directions; $\Omega = \alpha\, \Delta t/(\Delta x)2$ and $\Delta x = \Delta y = \Delta z$.]

Elapsed time, sec	Position from center along diagonal x = y = z	Exact analysis result, °C	Exponential finite difference results, °C N = 11, m = 4, Ω = 0.75 (a)	Pure explicit finite difference results, °C Ω = 0.15, N = 11 (a)	Method of Douglas Douglas finite difference results, °C Ω = 0.15, N = 11 (a)
0.09	0.0	0.893490	0.892237 (0.14)	0.889437 (0.45)	0.886760 (0.75)
	.5	.440712	.440650 (.014)	.435058 (1.28)	.439665 (.24)
	.9	.006491	.006484 (.11)	.006319 (2.65)	.006510 (-.29)
.15	0.0	.645469	.645209 (.04)	.640025 (.84)	.641484 (.62)
	.5	.253065	.253286 (-.09)	.250102 (1.17)	.252641 (.17)
	.9	.003015	.003022 (-.23)	.002970 (1.49)	.003023 (-.27)

[a]Accuracy percent.

TABLE VIII. - COMPARISON OF C.P.U. TIME ON TWO DIFFERENT MAINFRAMES FOR THREE DIFFERENT THREE-DIMENSIONAL FINITE DIFFERENCE METHODS

[One-hundred time steps for each method.]

Computer	Exponential[a] method, sec	Method of Douglas, sec	Pure-explicit method, sec
CRAY-XMP	0.2778	0.955	0.0627
IBM-3033	5.4	12.6	1.8

[a]Based on the number of sub-time intervals equal to 100.

TABLE IX. - COMPARISON OF EXPONENTIAL FINITE DIFFERENCE METHOD TO EXACT RESULTS OF BOUNDARY LAYER EQUATION [10] FOR THE VELOCITY PROFILE AT ONE DOWNSTREAM LOCATION.

[Distance downstream x = 500 cm, ν = 0.0072 cm^2/s.]

Distance perpendicular to plate, y (cm)	Exact result [10]	Exponential method result, N = 21 m = 8
1.0	0.17	0.17428
2.0	.34	.34643
3.0	.51	.51020
4.0	.65	.65658
5.0	.78	.77684
6.0	.87	.86636
7.0	.93	.92638
8.0	.96	.96265

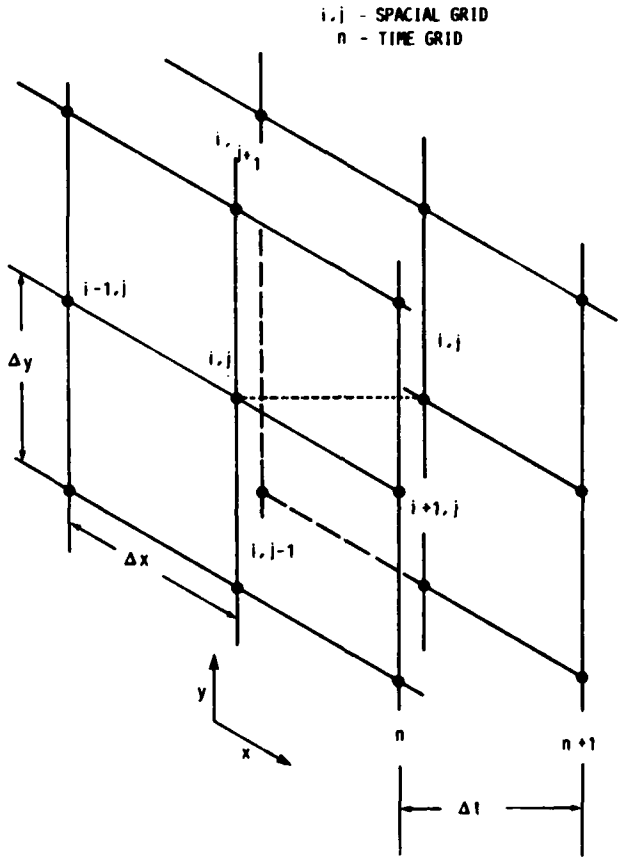


FIGURE 1.- COMPUTATIONAL GRID FOR EXPONENTIAL FINITE DIFFERENCE TECHNIQUE FOR 2-DIMENSIONAL CARTESIAN COORDINATES.

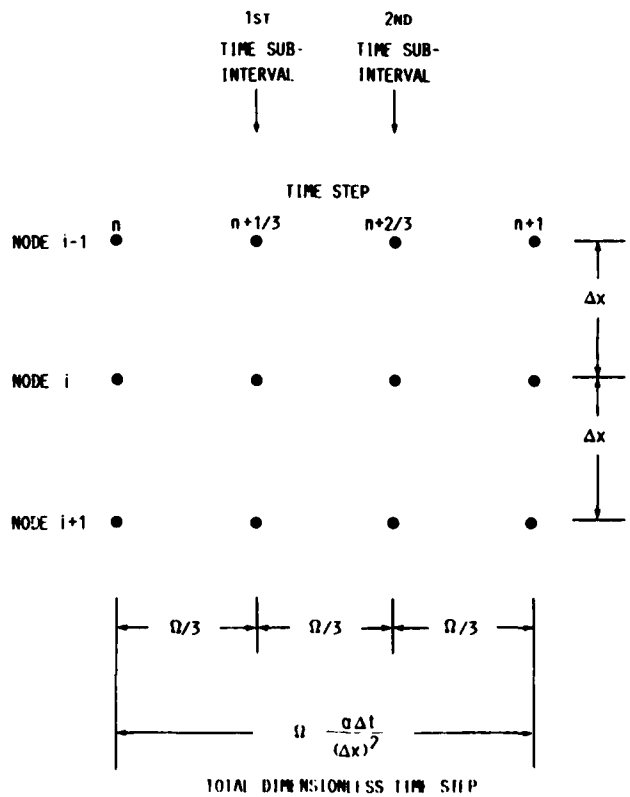


FIGURE 2. COMPUTATIONAL GRID FOR 2 TIME SUB INTERVALS (M = 2), CARTESIAN COORDINATES. (SHOWN FOR 1-DIMENSION).

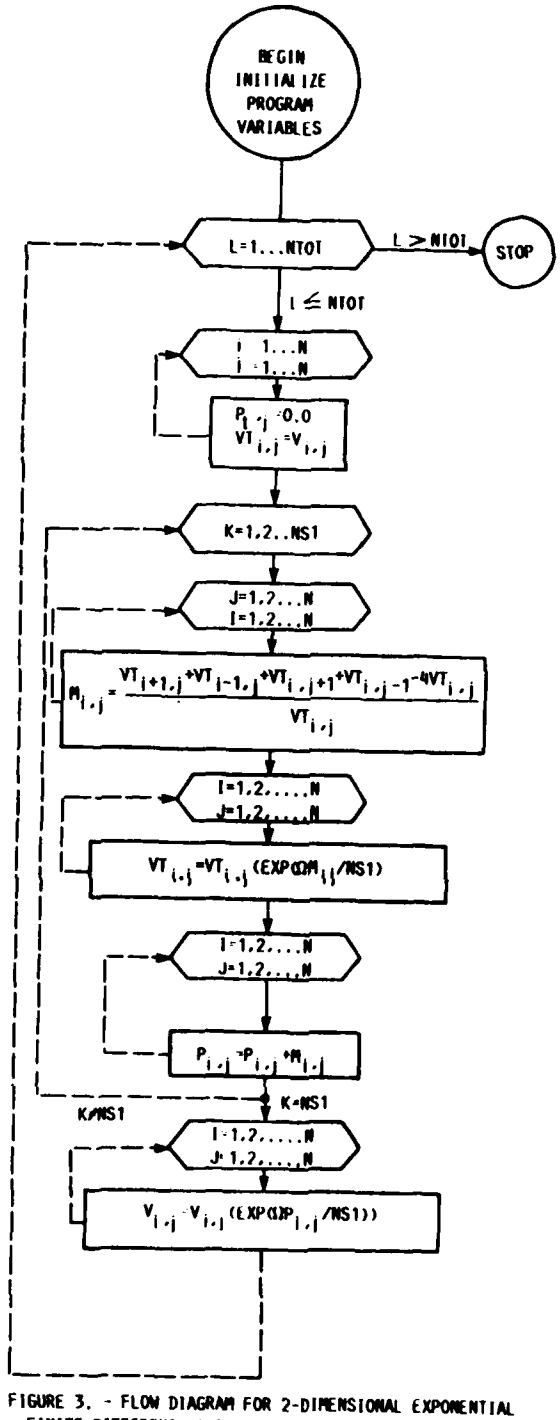


FIGURE 3. - FLOW DIAGRAM FOR 2-DIMENSIONAL EXPONENTIAL FINITE DIFFERENCE ALGORITHM.

NTOT - TOTAL NUMBER OF TIME STEPS
$P_{i,j}$ - SUM OF DIMENSIONLESS DRIVE NUMBERS
$VT_{i,j}$ - FIELD VARIABLE DURING TIME STEP
$M_{i,j}$ - DIMENSIONLESS DRIVE NUMBER
NS1 - NUMBER OF TIME SUB-INTERVALS
Ω - DIMENSIONLESS TIME $\frac{\alpha \Delta t}{(\Delta x)^2}$

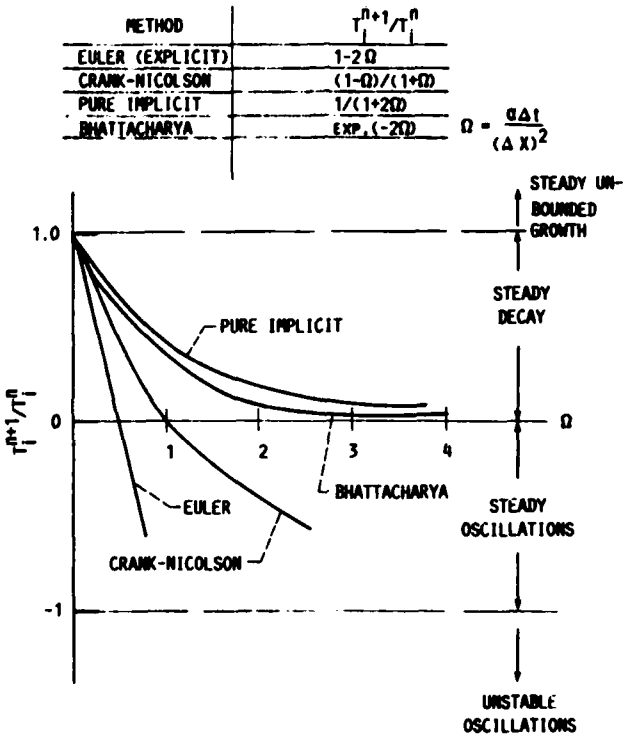


FIGURE 4. - EFFECT OF NONDIMENSIONAL TIME STEP SIZE ON 1 MODE MODEL SOLUTION. (FROM REF. [2]).

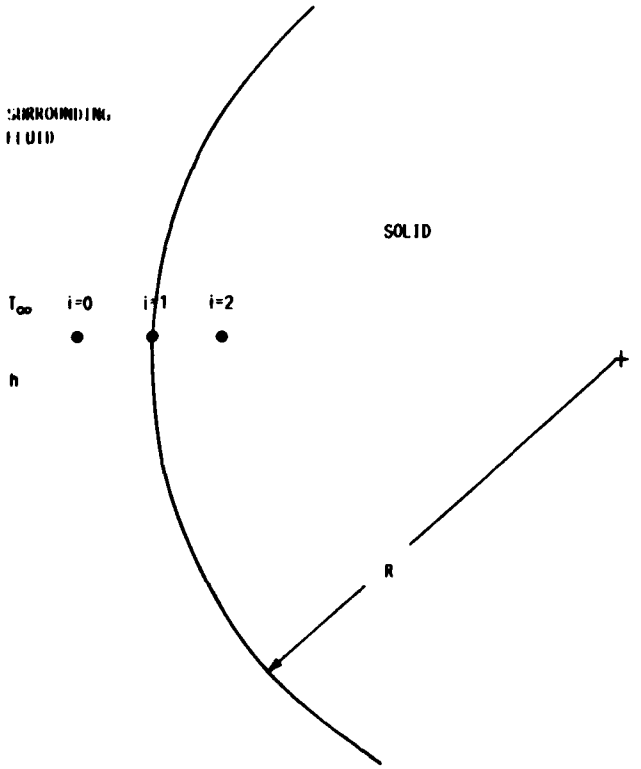
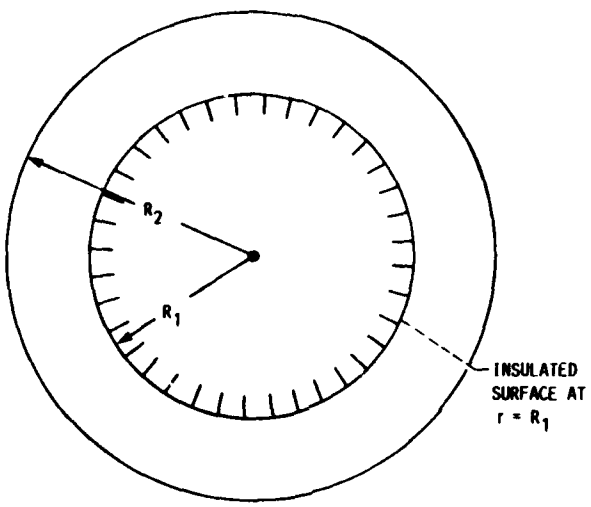


FIGURE 5. - EXPONENTIAL FINITE DIFFERENCE NODAL CONFIGURATION FOR 1-DIMENSIONAL HEAT TRANSFER WITH FINITE HEAT TRANSFER COEFFICIENT.



INITIAL CONDITION: $T(r,0) = 0$

BOUNDARY CONDITIONS: $T(R_2, t) = 1.0$

$$\frac{\partial T}{\partial r}(R_1,t) = 0$$

R_1 10.0 IN. R_2 = 19.0 IN.

FIGURE 6. - PROBLEM CONDITIONS FOR COMPARISON OF EXPONENTIAL FINITE DIFFERENCE TECHNIQUE TO CHARACTERISTIC PROBLEM SOLUTION.

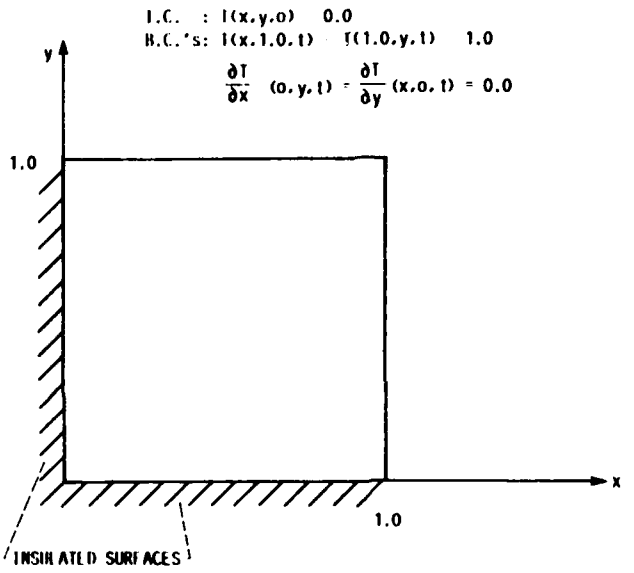


FIGURE 7. - PROBLEM DESCRIPTION FOR 2 DIMENSIONAL COMPARISON OF EXPONENTIAL FINITE DIFFERENCE METHOD TO THE ALTERNATING DIRECTION IMPLICIT (ADI) METHOD.

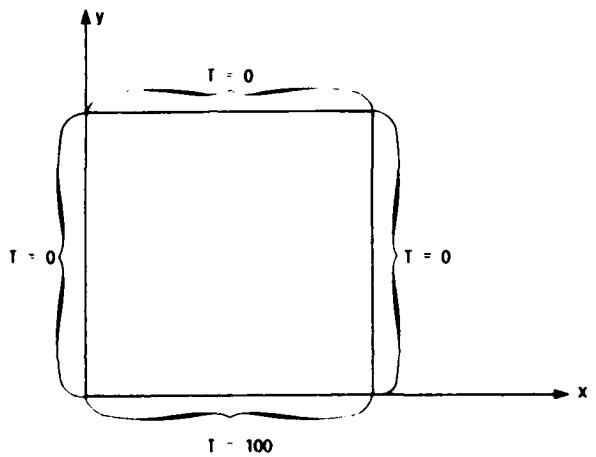
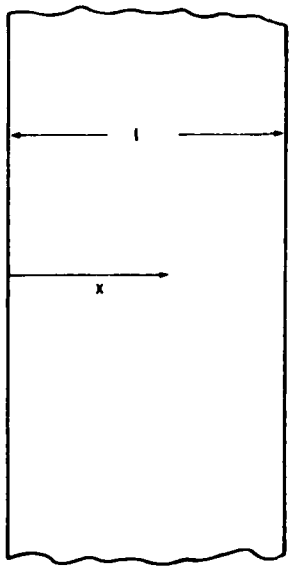


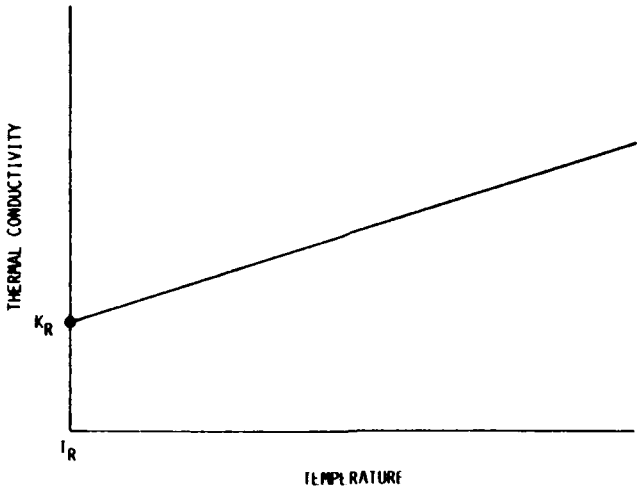
FIGURE 8. - PROBLEM SKETCH FOR COMPARISON OF EXPONTENTIAL FINITE DIFFERENCE SOLUTION TO THE GAUSS-SEIDEL ITERATIVE METHOD FOR THE SOLUTION OF LAPLACES EQUATION IN 2-DIMENSIONS.



T(x,0) 100.0
T(0,t) T(l,t) 0.0

$K(T) = K_R + \beta K_R T$

(A) ONE-DIMENSIONAL PROBLEM WITH VARYING THERMAL CONDUCTIVITY.



(B) LINEAR RELATIONSHIP BETWEEN CONDUCTIVITY AND TEMPERATURE.

FIGURE 9. SKETCHES SHOWING PROBLEM STATEMENT FOR TEMPERATURE VARYING THERMAL CONDUCTIVITY.

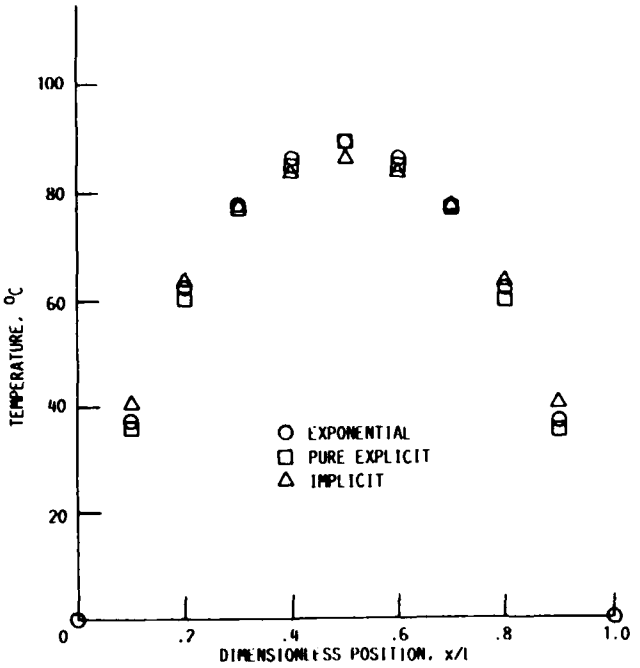
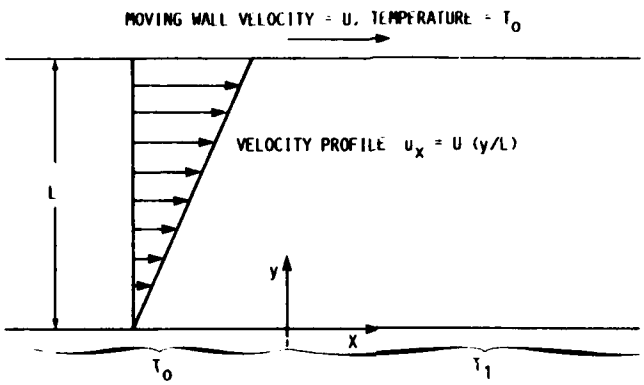


FIGURE 10. - COMPARISON OF METHODS FOR TEMPERATURE-VARYING CONDUCTIVITY, SHOWING TEMPERATURE FIELD AT t = 0.02 sec. $K(T) = K_R(1+\beta T)$, WHERE K_R = 1.0; β = 0.01; T(x,0) = 100; T(0,t) = T(L,t) = 0; α = 1.
EXPONENTIAL : m = 4; Ω = 0.5; 20 TIME STEPS.
PURE IMPLICIT: Ω = 0.25; 8 TIME STEPS.



BOUNDARY CONDITIONS: $T(y,o) = T_o \qquad x < o$

$T(o,x) = T_1 \qquad x \geq 0$

FIGURE 11. - SKETCH SHOWING CONDITIONS FOR DEVELOPING TEMPERATURE FIELD IN LAMINAR COUETTE FLOW.

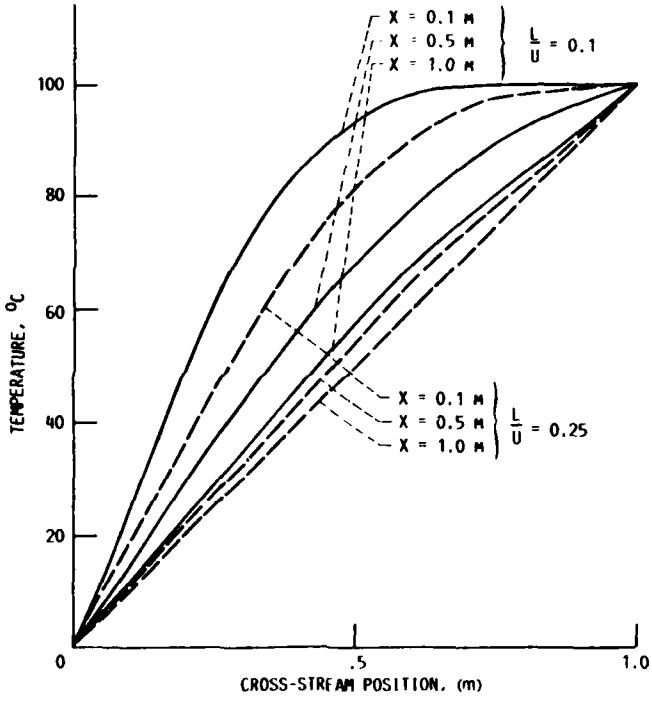
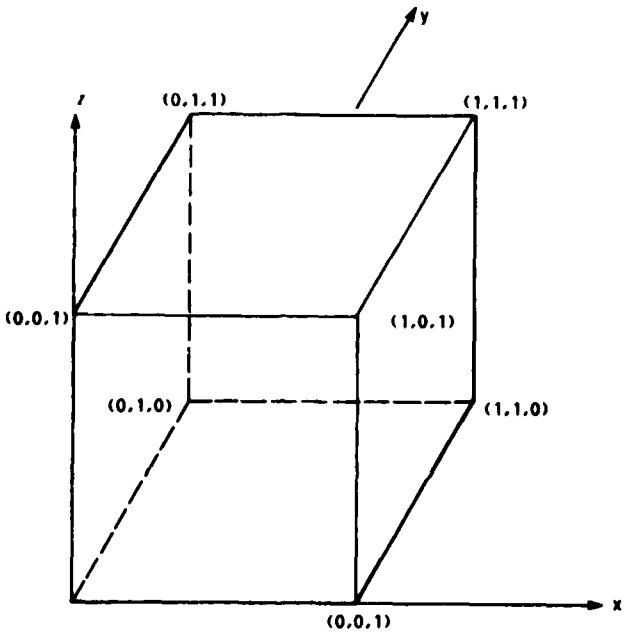


FIGURE 12. - DEVELOPING TEMPERATURE FIELD IN LAMINAR COUETTE FLOW, SHOWING THE EFFECT OF FLUID VELOCITY "U"; (L = 1.0 M).

3-DIMENSIONAL UNSTEADY STATE CONDUCTION



INITIAL CONDITION: $T(x,y,z,0) = T_0 = 1$

BOUNDARY CONDITIONS: $t > 0$

$$\frac{\partial T}{\partial x}(0,y,z,t) \quad \frac{\partial T}{\partial y}(x,0,z,t) \quad \frac{\partial T}{\partial z}(x,y,0,t) = 0$$

$$T(1,y,z,t) \quad T(x,1,z,t) \quad T(x,y,1,t) = 0$$

FIGURE 13. - BOUNDARY AND INITIAL CONDITIONS FOR THREE DIMENSIONAL UNSTEADY STATE CONDUCTION HEAT TRANSFER.

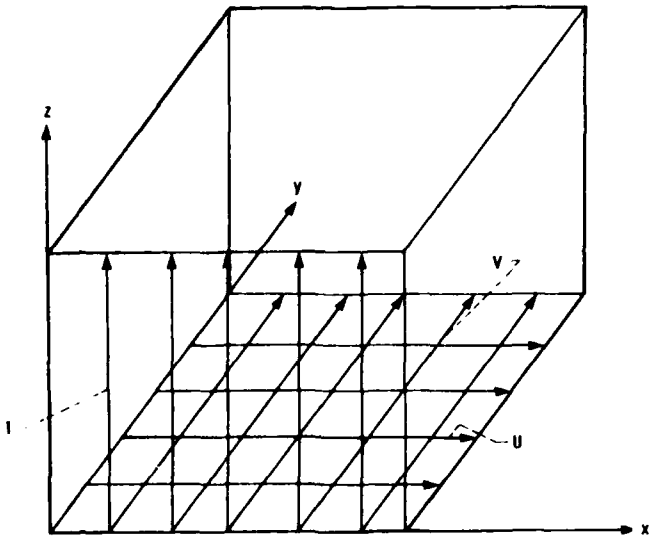


FIGURE 14. - METHOD OF DOUGLAS SHOWING THE PROCEDURE USED TO SWEEP IN THE SUCCESSIVE COORDINATE DIRECTIONS.

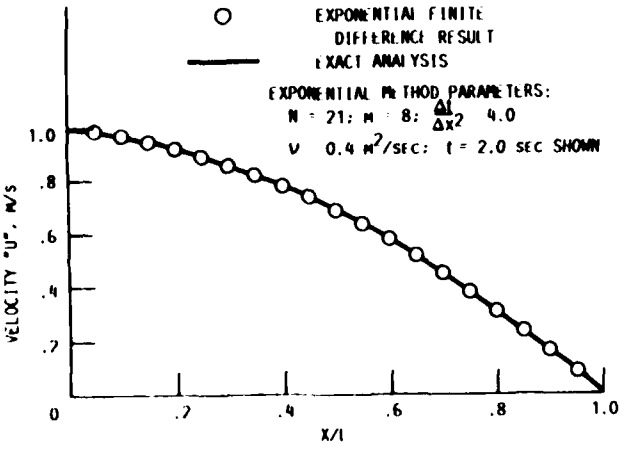


FIGURE 15. - COMPARISON OF STEADY STATE SOLUTIONS COMPARING THE EXACT RESULTS TO THE EXPONENTIAL FINITE DIFFERENCE SOLUTION U(0,t) 1.0; U(1,t) 0.0.

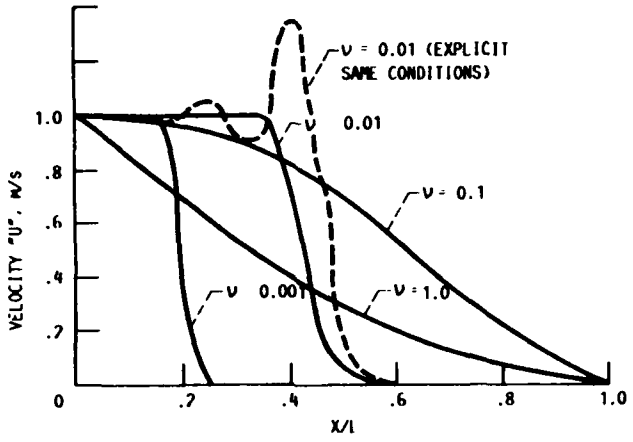
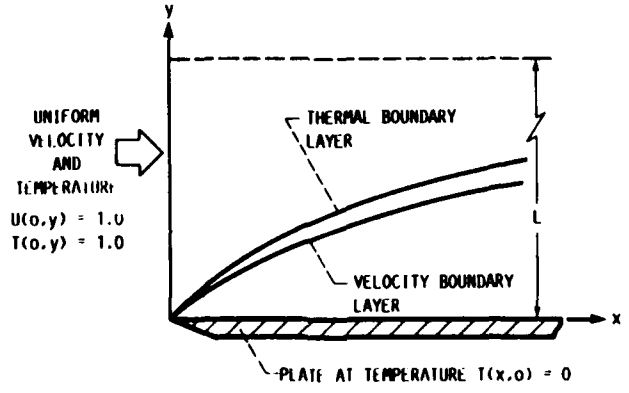


FIGURE 16. EXPONENTIAL FINITE DIFFERENCE RESULTS FOR VARYING KINEMATIC VISCOSITY. ALL VELOCITIES ARE SHOWN FOR t = 1.0 SECONDS WITH: N = 21, M = 8, $\frac{\Delta t}{(\Delta X)^2} = 4.0$.
U(o,t) = 1.0
U(L,t) = 0.0



OTHER CONDITIONS: U(x,o) 0; V(x,o) 0
V(x,t) 0

FIGURE 17. BOUNDARY LAYER DEVELOPMENT ALONG A COOLED FLAT PLATE.

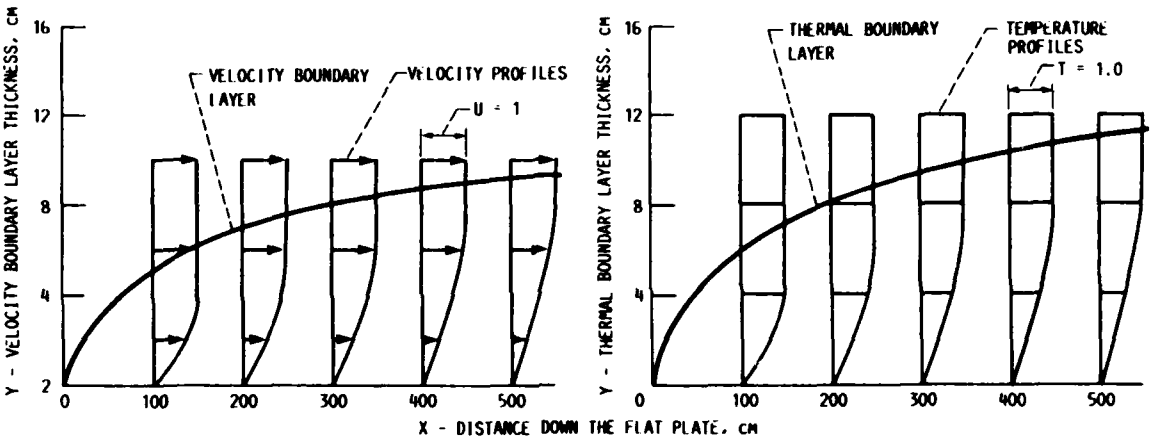


FIGURE 18. - EXPONENTIAL FINITE DIFFERENCE RESULTS FOR BOUNDARY LAYER EQUATIONS, WITH CONDITIONS U(o,y) = 1.0, U(x,o) 0, V(x,o) - 0, V(x,L) 0, T(x,o) = 0.0, T(o,y) = 1.0; ν = 0.0072 cm^2/sec; α = 0.01 cm^2/sec.

APPENDIX

This appendix contains all the computer programs mentioned in this report. A computer program variable list is also contained with a description of their use, and a program number to refer to the programs that they are contained in.

Each of these programs was written to be run in an interactive mode with the mainframe computer. The only cases run differently were for the three-dimensional unsteady state heat transfer cases that were run in batch mode on the Cray X-MP.

The program structure is as follows. A main program is used to describe the necessary parameters and for asserting the proper boundary conditions. The main program then calls the subroutine where the actual finite difference methods are exercised and the results are then printed.

COMPUTER PROGRAM LIST

Program number	Program name	Program function
1	SOURCE.EFDCYL	One-dimension, unsteady state, cylindrical coordinates, infinite and finite heat transfer coefficient
2	SOURCE.EFD2D	Two-dimensional Cartesian coordinates, unsteady state heat transfer
3	SOURCE.LAPLAC	Two-dimensional Laplace's equation
4	SOURCE.EFDVAR	One-dimensional unsteady state heat conduction, varying thermal conductivity, exponential finite difference method
5	SOURCE.EXPVAR	One-dimensional, unsteady state heat conduction, varying thermal conductivity, explicit finite difference method
6	SOURCE.IMPVAR	One-dimensional, unsteady state heat transfer, varying thermal conductivity, implicit finite difference method
7	SOURCE.COUE	One-dimensional, developing temperature field in laminar couette flow
8	SOURCE.EX3D	Exact analysis, three-dimensional heat transfer in a cube
9	SOURCE.EFD3D	Three-dimensional unsteady state heat transfer in a cube using exponential finite difference method
10	SOURCE.EXPL3D	Three-dimensional unsteady state heat transfer in a cube using explicit finite difference method
11	SOURCE.DOUGLA	Three-dimensional unsteady state heat transfer in a cube using the method of Douglas
12	SOURCE.BURGER	Exponential solution of nonlinear viscous Burger's equation
13	SOURCE.EXBURG	Pure explicit solution of nonlinear viscous Burger's equation
14	SOURCE.NONBOU	Exponential Method of solution for boundary layer equations for flow over a flat plate

COMPUTER PROGRAM VARIABLE LIST

Program variable name	Programs used in	Variable Description
N	1-14	Number of nodes
NS	1-12,14	Number of time sub intervals
NTOT	1-14	Total number of time or spacial steps
TSI	1-13	Dimensionless time step increment
T	1-7,9,10,12,13	Total elapsed time or spatial distance between steps
DL	1	Radial distance between adjacent nodes
R	1	Radial length
IPR	1-14	Number of steps between output of results
NB	1	Heat transfer boundary condition flag
V	1-14	Dependent variable
HK	1	Convection heat transfer coefficient divided by thermal conductivity
VM	1	Dependent variable value outside of solid in the surrounding medium
M	1-10,12	Dimensionless drive number
P	1-10,12	Sum of the drive numbers
VT	1-14	Dependent variable value during sub-time interval
B	1	Biot number
THE	1-3,12-14	Variable used for the output of results
TIME	1-6,8-13	Total elapsed time of the solution at the current output
ITMAX	1-14	Output counter
ETI	3	Accuracy desired in solution of Laplace's equation

Program variable name	Program numbers used in	Description
DELV	3	Difference in dependent variable value from one time step to the next
ETA	3	Sum of the absolute value of the differences found in DELV
KR	4-6	Reference thermal conductivity
BETA	4-6	Slope of thermal conductivity variation with temperature
KO,K	4	Thermal conductivity at the total time step interval or sub-time interval respectively
THE	4	Kirchoff transformation variable (used in exponential finite difference program with varying thermal conductivity)
KAPPA	4	All values known from the last time step increment and used to solve the quadratic equation that results in the exponential finite difference solution with temperature varying thermal conductivity
DERSQR	5	Absolute value of velocity difference found in evaluation of velocity gradient
OMEGA	6	Same as nondimensional time step
A,B,C,D	6,11	Coefficient used in tridiagonal matrix algorithm.
KAP,GAM	6	Variables used to determine A,B,C
BETA,GAMMA	6	Variables used in Thomas, tridiagonal algorithm
TS	6	Same as total elapsed time
T	6	Dependent variable
V	6	Solution vector tri-diagonal algorithm
SP	7	Maximum width divided by maximum velocity
FL	7	Parameter based on position in flow
DIST	7	Serves same function as time for unsteady state problem

Program variable name	Program numbers used in	Description
PI,PI2,PI3	8	π, π^2, π^3
NODES	8	Same an N
TO	8	Initial temperature
TI	8	Surface temperature, t > 0
ALFA	8	Thermal diffusivity
TNEW	8	Exact temperature at a x, y, and z location after elapsed time t has occurred
T	11	Dependent variable
DELX,DELY,DELZ	11	Part of the central difference operator
U,V	11	Variables used to sweep preliminary solution in the x then y directions respectively
M	11	Used as an array to contain the known quantities used in the tridiagonal algorithm
RNU	12-14	Kinematic viscosity
DX	14	Step length in flat plate direction
RAL	14	Thermal diffusivity
YMAX	14	Maximum distance perpendicular to flat plate
DY	14	Step length perpendicular to the flat plate
U,V	14	Dependent variables (velocity) in x and y directions respectively
T	14	Temperature field variable
MU,MT	14	Drive numbers for velocity in x-direction and temperature field
PU,PT	14	Sum of drive numbers for MU, MT
UT,TT,VT	14	x-direction velocity, temperature, and y-direction velocity on subinterval
UT1	14	Temporary U-direction velocity field
TS,TS1	14	Dimensionless time step for temperature and x-direction velocity respectively

```
C
C     WRITTEN BY R.F. HANDSCHUH
C
C     SOURCE.EFDCYL
C
C     ***** PROGRAM #1 *****
C
C
C     THIS PROGRAM IS TO BE USED AS THE STARTING POINT FOR INVESTIGATING
C     THE EXPONENTIAL FINITE DIFFERENCE ALGORYTHM.  THIS METHOD WAS INTRODUCED
C     BY M.C. BHATTACHARYA.  THIS SOURCE CODE IS FOR CYLINDRICAL
C     COORDINATES, UNSTEADY-STATE HEAT CONDUCTION, 1 DIMENSION.
C
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 V(100),R(100)
C
C     INPUT THE PROGRAM DATA
C
      WRITE(6,15)
 15   FORMAT(1X,'NUMBER OF NODES=N   I3'/)
      READ(9,10)N
 10   FORMAT(I3)
      WRITE(6,12)
 12   FORMAT(1X,'NUMBER OF TIME SUB INTERVALS= NS  I3')
      READ(6,13)NS
 13   FORMAT(I3)
      WRITE(6,16)
 16   FORMAT(1X,'TOTAL NUMBER OF TIME STEPS= NTOT   I3')
      READ(9,21)NTOT
 21   FORMAT(I3)
      WRITE(6,24)
 24   FORMAT(1X,'INPUT TIME*THERMAL DIFFUSIVITY / RAD INT SQUARED  F5.3')
      READ(9,25)TSI
 25   FORMAT(F5.3)
      WRITE(6,22)
 22   FORMAT(1X,'TOTAL TIME OF ONE TIME STEP= T F7.5')
      READ(9,23)T
 23   FORMAT(F7.4)
      WRITE(6,26)
 26   FORMAT(1X,'INPUT RADIAL INTERVAL LENGTH=DL F5.3')
      READ(9,27)DL
 27   FORMAT(F5.3)
      R(1)=1.
      DO 28 I=2,N
      IM1=I-1
 28   R(I)=R(IM1)-DL
      WRITE(6,32)
 32   FORMAT(1X,'INPUT NUMBER OF TIME STEPS BEFORE PRINTING  I3')
      READ(9,33)IPR
 33   FORMAT(I3)
C
C     DETERMINE THE TYPE OF BOUNDRY CONDITION, THEN SET VALUES
C
```

```
      WRITE(6,14)
 14   FORMAT(1X,'INPUT HEAT TRANSFER B.C.   0 - INFINITE  1 - FINITE'/)
      READ(9,17)NB
 17   FORMAT(I1)
      IF(NB.EQ.1) GO TO 100
      V(1)=0.
      DO 30 I=2,N
 30   V(I)=1.0
C
C     CALL EXP FIN DIF FOR INFINITE HEAT TRANSFER COEFFICENT
C
      CALL EFDIHC(N,NS,NTOT,TSI,V,T,R,DL,IPR)
      GO TO 101
 100  CONTINUE
      WRITE(6,31)
 31   FORMAT(1X,'INPUT HEAT COEF / TERM COND  F5.3')
      READ(9,34)HK
 34   FORMAT(F5.3)
      DO 40 I=1,N
 40   V(I)=1.
      VM=0.
C
C     CALL EXP FIN DIF FOR FINITE HEAT TRANSFER COEFFICENT
C
      CALL EFDFHC(HK,N,VM,DL,NS,V,NTOT,TSI,T,R,IPR)
 101  CONTINUE
      STOP
      END
C
C     SUBROUTINE EFDIHC
C
      SUBROUTINE EFDIHC(N,NS,NTOT,TSI,V,T,R,DL,IPR)
C
C     FOR INFINITE HEAT TRANSFER COEFFICENT
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 VT(100),V(100),M(100),P(100),R(100),THE(100)
      TS=TSI/DFLOAT(NS+1)
      WRITE(6,21)(R(I),I=1,N)
 21   FORMAT(1X,11(F6.3,2X))
      WRITE(6,22)DL,TSI
 22   FORMAT(1X,2(F6.3,2X))
      N1=N-1
      NS1=NS+1
C
C     BEGIN MAIN TIME STEP LOOP
C
      DO 20 L=1,NTOT
C
C     ZERO DRIVE NUMBERS AND SET TEMPORARY VARIABLES EQUAL TO THE
C     LAST TOTAL TIME STEP VALUES
C
      DO 15 I=1,N
 15   P(I)=0.
```

```
      DO 10 I=1,N
 10   VT(I)=V(I)
C
C     SUB TIME INTERVAL
C
      DO 30 K=1,NS1
C
C     CALCULATE THE DRIVE NUMBERS
C
      DO 40 I=2,N1
      IM1=I-1
      IP1=I+1
      M(I)=(2.*VT(I)-VT(IM1)-VT(IP1))/VT(I)
 40   M(I)=M(I)+DL*(VT(IP1)-VT(IM1))/(VT(I)*2.*R(I))
      M(N)=2.*(VT(N)-VT(N1))/VT(N)
C
C     CALCULATE THE DEPENDENT VARIABLE ON THE SUB-INTERVAL LEVEL
C
      DO 50 I1=2,N
 50   VT(I1)=VT(I1)*DEXP(-TS*M(I1))
C
C     SUM THE DRIVE NUMBERS
C
      DO 60 I=2,N
 60   P(I)=P(I)+M(I)
 30   CONTINUE
C
C     CALCULATE THE DEPENDENT VARIABLE ON THE NEXT COMPLETE TIME STEP
C
      DO 70 I=1,N
      V(I)=V(I)*DEXP(-TS*P(I))
 70   CONTINUE
      ITMAX=ITMAX+1
C
C     PRINT THE RESULTS
C
      IF(ITMAX.LT.IPR)GO TO 20
      DO 71 I=1,N
 71   THE(I)=V(I)
      ITMAX=0
      WRITE(6,5)
 5    FORMAT(/)
      WRITE(6,31)L
 31   FORMAT(1X,'TIME STEP NUMBER=',I3)
      TIME=T*DFLOAT(L)
      WRITE(6,32)TIME
 32   FORMAT(1X,'ELAPSED TIME=',F10.4,' SECONDS')
      IF(N.GT.11)N21=N/2
      IF(N.GT.11)GO TO 81
      WRITE(6,82)(THE(I),I=1,N)
 82   FORMAT(11(2X,F8.6))
      GO TO 84
 81   CONTINUE
      WRITE(6,82)(THE(I),I=1,N21)
```

```
      NNS=N21+1
      WRITE(6,82)(THE(I),I=NNS,N)
 84   CONTINUE
 20   CONTINUE
      RETURN
      END
C
C     SUBROUTINE EFDFHC
C
      SUBROUTINE EFDFHC(HK,N,VM,DL,NS,V,NTOT,TSI,T,R,IPR)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 VT(100),V(100),M(100),P(100),R(100),THE(100)
      B=HK*DL
      TS=TSI/DFLOAT(NS+1)
      NS1=NS+1
      N1=N-1
      N2=N-2
C
C     BEGIN THE TOTAL TIME STEP LOOP
C
      DO 20 L=1,NTOT
C
C     ZERO THE DRIVE NUMBERS AND SET THE TEMPOARY DEPENDET VARIABLES
C     EQUAL TO THE LAST COMPLETE STEP VALUES
C
      DO 15 I=1,N
 15   P(I)=0.
      DO 10 I=1,N
 10   VT(I)=V(I)
C
C     SUB TIME INTERVAL
C
      DO 30 K=1,NS1
C
C     CALCULATE THE DRIVE NUMBERS
C
      M(1)=-(2.*VT(2)-(2.+2.*B)*VT(1)+2.*B*VM)/VT(1)
      M(1)=M(1)-DL*B*(VT(1)-VM)/(R(1)*VT(1))
      DO 40 I=2,N1
      IM1=I-1
      IP1=I+1
      M(I)=(2.*VT(I)-VT(IM1)-VT(IP1))/VT(I)
 40   M(I)=M(I)+DL*(VT(IP1)-VT(IM1))/(VT(I)*2.*R(I))
      M(N)=2.*(VT(N)-VT(N1))/VT(N)
C
C     CALCULATE THE DEPENDENT VARIABLE ON THE SUB-INTERVAL LEVEL
C
      DO 50 I1=1,N
 50   VT(I1)=VT(I1)*DEXP(-TS*M(I1))
C
C     SUM THE DRIVE NUMBERS
C
      DO 60 I=1,N
 60   P(I)=P(I)+M(I)
```

```
 30   CONTINUE
C
C     CALCULATE THE DEPENDENT VARIABLE ON THE NEXT COMPLETE TIME STEP
C
      DO 70 I=1,N
      V(I)=V(I)*DEXP(-TS*P(I))
 70   CONTINUE
      ITMAX=ITMAX+1
      IF(ITMAX.LT.IPR)GO TO 20
C
C     PRINT THE RESULTS
C
      DO 71 I=1,N
 71   THE(I)=V(I)
      ITMAX=0
      WRITE(6,5)
 5    FORMAT(/)
      WRITE(6,31)L
 31   FORMAT(1X,'TIME STEP NUMBER=',I3)
      TIME=T*DFLOAT(L)
      WRITE(6,32)TIME
 32   FORMAT(1X,'ELAPSED TIME=',F10.4,' SECONDS')
      IF(N.GT.11)N21=N/2
      IF(N.GT.11)GO TO 81
      WRITE(6,82)(THE(I),I=1,N)
 82   FORMAT(11(2X,F8.6))
      GO TO 84
 81   CONTINUE
      WRITE(6,82)(THE(I),I=1,N21)
      NNS=N21+1
      WRITE(6,82)(THE(I),I=NNS,N)
 84   CONTINUE
 20   CONTINUE
      RETURN
      END
```

```
C
C     SOURCE.EFD2D
C
C     WRITTEN BY R.F. HANDSCHUH
C
C     ***** PROGRAM #2 *****
C
C     THIS PROGRAM IS FOR 2-DIMENSIONAL CARTESIAN COORDINATES
C     UNSTEADY STATE HEAT TRANSFER.  THE METHOD OF SOLUTION IS THE
C     EXPONENTIAL FINITE DIFFERENCE ALGORYTHM.  THIS PARTICULAR PROGRAM IS
C     FOR INFINITE HEAT TRANSFER COEFFICENT AT THE EXPOSED SURFACES
C     AT X=Y=1.0    FOR X=Y=0 THE SURFACE IS CONSIDERED TO BE
C     PERFECTLY INSULATED.
C
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 V(25,25)
C
C     INPUT PROGRAM DATA
C
      WRITE(6,15)
 15   FORMAT(1X,'NUMBER OF NODES=N   I3'/)
      READ(9,10)N
 10   FORMAT(I3)
      WRITE(6,12)
 12   FORMAT(1X,'NUMBER OF TIME SUB INTERVALS= NS  I3')
      READ(9,13)NS
 13   FORMAT(I3)
      WRITE(6,16)
 16   FORMAT(1X,'TOTAL NUMBER OF TIME STEPS= NTOT   I3')
      READ(9,21)NTOT
 21   FORMAT(I3)
      WRITE(6,24)
 24   FORMAT(1X,'INPUT TIME*THERMAL DIFFUSIVITY / LENGTH SQUARED  F5.3')
      READ(9,25)TSI
 25   FORMAT(F5.3)
      WRITE(6,22)
 22   FORMAT(1X,'TOTAL TIME OF ONE TIME STEP= T F6.4')
      READ(9,23)T
 23   FORMAT(F6.4)
      WRITE(6,26)
 26   FORMAT(1X,'NUMBER OF TIME STEPS BEFORE PRINTING= I3')
      READ(9,27)IPR
 27   FORMAT(I3)
      WRITE(6,250)N,NS,NTOT
 250  FORMAT(1X,'# OF NODES=',I3,2X,'# OF SUB-TIME-INT=',I3,2X,       -
     *'# OF TIME STEPS=',I3)
      WRITE(6,251)TSI,T
 251  FORMAT(1X,'(TIME*THER DIFF)/LENGTH SQUARED='F5.3,2X,            -
     *'TIME STEP LENGTH=',F6.4/)
      N1=N-1
C
C     INITIALIZE THE BOUNDRY CONDITIONS
C
```

```
      DO 30 I=1,N1
      DO 30 J=1,N1
 30   V(I,J)=1.0
C
      I=N
      DO 50 J=1,N
 50   V(I,J)=0.0
      J=N
      DO 51 I=1,N
 51   V(I,J)=0.0
C
C     CALL EXP FIN DIF FOR INFINITE HEAT TRANSFER COEFFICENT
C
      CALL EFDIHC(N,NS,NTOT,TSI,V,T,IPR)
      STOP
      END
C
C     SUBROUTINE EFDIHC
C
      SUBROUTINE EFDIHC(N,NS,NTOT,TSI,V,T,IPR)
C
C     FOR INFINITE HEAT TRANSFER COEFFICENT
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 VT(25,25),V(25,25),M(25,25),P(25,25),THE(25,25)
C
C     PRINT HEADING
C
      WRITE(6,222)
 222  FORMAT(1X,'************   SOURCE.EFD2D   ************'//)
      TS=TSI/DFLOAT(NS+1)
      N1=N-1
      NS1=NS+1
C
C     BEGIN MAIN TIME STEP LOOP
C
      DO 20 L=1,NTOT
C
C     ZERO THE DRIVE NUMBERS AND SET TEMPORARY DEPENDENT VARIABLE
C     EQUAL TO THE LAST FULL TIME STEP VALUE
C
      DO 15 J=1,N
      DO 15 I=1,N
 15   P(I,J)=0.
      DO 10 J=1,N
      DO 10 I=1,N
 10   VT(I,J)=V(I,J)
C
C     SUB TIME INTERVAL
C
      DO 30 K=1,NS1
C
C     CALCULATE THE DRIVE NUMBERS
C
```

```
      DO 41 J=2,N1
      JM1=J-1
      JP1=J+1
      DO 40 I=2,N1
      IM1=I-1
      IP1=I+1
      M(I,J)=(VT(IP1,J)+VT(IM1,J)+VT(I,JP1)+VT(I,JM1)-4.*VT(I,J))
 40   M(I,J)=M(I,J)/VT(I,J)
 41   CONTINUE
C
C     INSULATED BOUNDRY ALONG X-AXIS
C
      J=1
      JP1=J+1
      DO 42 I=2,N1
      IP1=I+1
      IM1=I-1
 42   M(I,J)=(VT(IP1,J)+VT(IM1,J)+2.*VT(I,JP1)-4.*VT(I,J))/VT(I,J)
C
C     INSULATED BOUNDRY ALONG Y-AXIS
C
      I=1
      IP1=I+1
      DO 43 J=2,N1
      JP1=J+1
      JM1=J-1
 43   M(I,J)=(2.*VT(IP1,J)+VT(I,JP1)+VT(I,JM1)-4.*VT(I,J))/VT(I,J)
C
C     CORNER AT ORIGIN
C
      M(1,1)=(2.*VT(1,2)+2.*VT(2,1)-4.*VT(1,1))/VT(1,1)
C
C     CALCULATE THE DEPENDENT VARIABLE ON THE SUB-INTERVAL LEVEL
C
      DO 50 I1=1,N1
      DO 50 J1=1,N1
 50   VT(I1,J1)=VT(I1,J1)*DEXP(TS*M(I1,J1))
C
C     SUM THE DRIVE NUMBERS
C
      DO 60 I=1,N1
      DO 60 J=1,N1
 60   P(I,J)=P(I,J)+M(I,J)
 30   CONTINUE
C
C     CALCULATE THE DEPENDENT VARIABLE AT THE NEXT COMPLETE TIME STEP
C
      DO 70 J=1,N
      DO 70 I=1,N
      V(I,J)=V(I,J)*DEXP(TS*P(I,J))
 70   CONTINUE
      ITMAX=ITMAX+1
      IF(ITMAX.LT.IPR)GO TO 20
C
```

```
C     PRINT THE RESULTS
C
      WRITE(6,5)
 5    FORMAT(/)
      WRITE(6,31)L
 31   FORMAT(1X,'TIME STEP NUMBER=',I3/)
      TIME=T*DFLOAT(L)
      WRITE(6,32)TIME
 32   FORMAT(5X,'ELAPSED TIME=',F10.4,'SECONDS'/)
      DO 71 I=1,N
      DO 71 J=1,N
 71   THE(I,J)=1.0-V(I,J)
      IF(N.GT.11)GO TO 58
      DO 59 J=1,N
      WRITE(6,82)(THE(I,J),I=1,N)
 82   FORMAT(11(2X,F8.6))
 59   CONTINUE
      GO TO 54
 58   CONTINUE
      DO 57 J=1,N
      WRITE(6,56)(THE(I,J),I=1,11)
 56   FORMAT(11(2X,F8.6))
      WRITE(6,56)(THE(I,J),I=12,N)
 57   CONTINUE
 54   CONTINUE
      ITMAX=0
 20   CONTINUE
      RETURN
      END
```

```
C
C     SOURCE.LAPLAC
C
C     WRITTEN BY R.F. HANDSCHUH
C
C     ***** PROGRAM #3 *****
C
C     THIS PROGRAM IS TO BE USED TO SOLVE THE LAPLACE'S EQUATION
C     USING THE EXPONENTIAL FINITE DIFFERENCE METHOD.
C
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 V(25,25)
C
C     INPUT PROGRAM DATA
C
      WRITE(6,15)
 15   FORMAT(1X,'NUMBER OF NODES=N   I3'/)
      READ(9,10)N
 10   FORMAT(I3)
      WRITE(6,12)
 12   FORMAT(1X,'NUMBER OF TIME SUB INTERVALS= NS  I3')
      READ(9,13)NS
 13   FORMAT(I3)
      WRITE(6,16)
 16   FORMAT(1X,'TOTAL NUMBER OF TIME STEPS= NTOT   I3')
      READ(9,21)NTOT
 21   FORMAT(I3)
      WRITE(6,24)
 24   FORMAT(1X,'INPUT TIME / LENGTH SQUARED  F5.3')
      READ(9,25)TSI
 25   FORMAT(F5.3)
      WRITE(6,22)
 22   FORMAT(1X,'TOTAL TIME OF ONE TIME STEP= T F7.6')
      READ(9,23)T
 23   FORMAT(F7.6)
      WRITE(6,26)
 26   FORMAT(1X,'NUMBER OF TIME STEPS BEFORE PRINTING= I3')
      READ(9,27)IPR
 27   FORMAT(I3)
      WRITE(6,31)
 31   FORMAT(1X,'INPUT ACCURACY DESIRED= F7.6')
      READ(9,32)ET1
 32   FORMAT(F7.6)
      N1=N-1
C
C     INITIALIZE THE BOUNDRY CONDITIONS
C
      DO 30 I=2,N1
      DO 30 J=2,N1
 30   V(I,J)=100.
C
      I=N
      DO 50 J=1,N
```

```
 50   V(I,J)=0.0
      J=N
      DO 51 I=1,N
 51   V(I,J)=0.0
C
C
      J=1
      DO 53 I=2,N1
 53   V(I,J)=0.0
C
C
      I=1
      DO 52 J=1,N
 52   V(I,J)=100.0
C
C
C     CALL EXP FIN DIF  FOR LAPLACE EQUATION
C
      CALL EFDIHC(N,NS,NTOT,TSI,V,T,IPR,ET1)
      STOP
      END
C
C     SUBROUTINE EFDIHC
C
      SUBROUTINE EFDIHC(N,NS,NTOT,TSI,V,T,IPR,ET1)
C
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 VT(25,25),V(25,25),M(25,25),P(25,25),THE(25,25)
      TS=TSI/DFLOAT(NS+1)
      N1=N-1
      NS1=NS+1
C
C     BEGIN MAIN STEP INCREMENT
C
      DO 20 L=1,NTOT
      ETA=0.0
C
C     ZERO THE SUM OF THE DRIVE NUMBERS
C
      DO 15 J=1,N
      DO 15 I=1,N
 15   P(I,J)=0.
C
C     SAVE THE DEPENDENT VALUES FROM THE LAST TOTAL TIME STEP
C
      DO 10 J=1,N
      DO 10 I=1,N
      DELV=V(I,J)-VT(I,J)
      ETA=ETA+DABS(DELV)
 10   VT(I,J)=V(I,J)
      IF(L.LE.88) GO TO 107
      IF(ETA.LE.ET1)GO TO 100
 107  CONTINUE
```

```
C
C       SUB TIME INTERVAL
C
        DO 30 K=1,NS1
C
C       CALCULATE THE DRIVE NUMBERS
C
        DO 41 J=2,N1
        JM1=J-1
        JP1=J+1
        DO 40 I=2,N1
        IM1=I-1
        IP1=I+1
        M(I,J)=(VT(IP1,J)+VT(IM1,J)+VT(I,JP1)+VT(I,JM1)-4.*VT(I,J))
 40     M(I,J)=M(I,J)/VT(I,J)
 41     CONTINUE
C
C       CALCULATE THE DEPENDENT VARIABLE ON THE SUB-TIME INTERVAL
C
        DO 50 I1=2,N1
        DO 50 J1=2,N1
 50     VT(I1,J1)=VT(I1,J1)*DEXP(TS*M(I1,J1))
C
C       SUM THE DRIVE NUMBERS
C
        DO 60 I=1,N1
        DO 60 J=1,N1
 60     P(I,J)=P(I,J)+M(I,J)
 30     CONTINUE
C
C       CALCULATE THE DEPENDENT VARIABLE ON THE NEXT TOTAL STEP
C
        DO 70 J=1,N
        DO 70 I=1,N
        V(I,J)=V(I,J)*DEXP(TS*P(I,J))
 70     CONTINUE
        ITMAX=ITMAX+1
C
C       PRINT THE RESULTS
C
        IF(ITMAX.LT.IPR)GO TO 20
        WRITE(6,5)
 5      FORMAT(/)
        WRITE(6,31)L
 31     FORMAT(1X,'TIME STEP NUMBER=',I3/)
        TIME=T*DFLOAT(L)
        DO 71 I=1,N
        DO 71 J=1,N
 71     THE(I,J)=V(I,J)
        IF(N.GT.11)GO TO 58
        DO 59 J=1,N
        WRITE(6,82)(THE(I,J),I=1,N)
 82     FORMAT(11(2X,F8.4))
 59     CONTINUE
```

```
      GO TO 54
58    CONTINUE
      DO 57 J=1,N
      WRITE(6,56)(THE(I,J),I=1,11)
56    FORMAT(11(2X,F8.4))
      WRITE(6,56)(THE(I,J),I=12,N)
57    CONTINUE
54    CONTINUE
      ITMAX=0
20    CONTINUE
      GO TO 101
100   CONTINUE
      WRITE(6,103)L
103   FORMAT(2X'*****  CONVERGED RESULT  *****',5X,'ITERATION=',I4//)
      DO 102 J=1,N
      WRITE(6,56)(V(I,J),I=1,N)
102   CONTINUE
101   CONTINUE
      RETURN
      END
```

```
C
C     WRITTEN BY R.F. HANDSCHUH
C
C     ***** PROGRAM #4 *****
C
C     SOURCE.EFDVAR
C
C
C     THIS PROGRAM IS FOR THE SOLUTION OF THE DIFFUSION EQUATION
C     WITH VARYING THERMAL CONDUCTIVITY.  THE METHOD USED IS THE
C     EXPONENTIAL FINITE DIFFERENCE METHOD.
C
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 V(100),KR
C
C     INPUT PROGRAM DATA
C
      WRITE(6,15)
 15   FORMAT(1X,'NUMBER OF NODES=N   I3'/)
      READ(9,10)N
 10   FORMAT(I3)
      WRITE(6,12)
 12   FORMAT(1X,'NUMBER OF TIME SUB INTERVALS= NS  I3')
      READ(6,13)NS
 13   FORMAT(I3)
      WRITE(6,16)
 16   FORMAT(1X,'TOTAL NUMBER OF TIME STEPS= NTOT   I3')
      READ(9,21)NTOT
 21   FORMAT(I3)
      WRITE(6,24)
 24   FORMAT(1X,'INPUT TIME*THERMAL DIFFUSIVITY / LENGTH SQUARED  F5.3')
      READ(9,25)TSI
 25   FORMAT(F5.3)
      WRITE(6,22)
 22   FORMAT(1X,'TOTAL TIME OF ONE TIME STEP= T F7.5')
      READ(6,23)T
 23   FORMAT(F7.5)
      WRITE(6,14)
 14   FORMAT(1X,'INPUT REFERENCE THERMAL CONDUCTIVITY=F5.4')
      READ(9,17)KR
 17   FORMAT(F5.4)
      WRITE(6,26)
 26   FORMAT(1X,'INPUT THERMAL CONDUCTIVITY SLOPE VALUE= F5.4')
      READ(9,27)BETA
 27   FORMAT(F5.4)
C
C     INITIALIZE THE BOUNDRY CONDITIONS
C
      V(1)=0.
      V(N)=0.
      N1=N-1
      DO 30 I=2,N1
 30   V(I)=100.
```

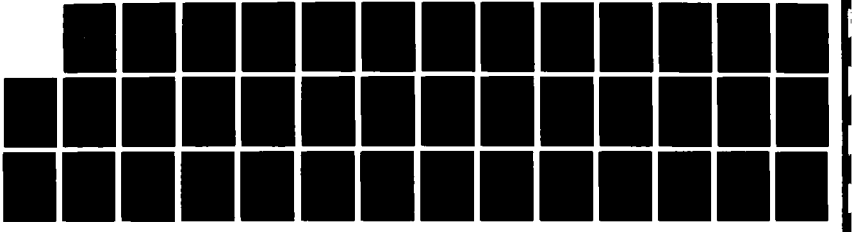
```
C
C     CALL EXP FIN DIF FOR INFINITE HEAT TRANSFER COEFFICENT
C
      CALL EFDIHC(N,NS,NTOT,TSI,V,T,KR,BETA)
      STOP
      END
C
C     SUBROUTINE EFDIHC
C
      SUBROUTINE EFDIHC(N,NS,NTOT,TSI,V,T,KR,BETA)
C
C     FOR INFINITE HEAT TRANSFER COEFFICENT
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 VT(100),V(100),M(100),P(100)
      REAL*8 KR,K(100),K0(100),THE(100),KAPPA(100)
      TS=TSI/DFLOAT(NS+1)
      N1=N-1
      NS1=NS+1
C
C     PRINT HEADING
C
      WRITE(6,5)
      WRITE(6,100)
  100 FORMAT(1X,'*****     SOURCE.EFDVAR     *****'/)
C
C     BEGIN MAIN TIME STEP LOOP
C
      DO 20 L=1,NTOT
C
C     ZERO THE SUM OF THE DRIVE NUMBERS
C
      DO 15 I=1,N
   15 P(I)=0.
C
C     SET VARIABLES EQUAL TO THE LAST STEPS VALUES
C
      DO 10 I=1,N
   10 VT(I)=V(I)
      DO 11 I=1,N
   11 K0(I)=KR+BETA*KR*V(I)
C
C     SUB TIME INTERVAL
C
      DO 30 KK=1,NS1
      DO 35 I=1,N
      K(I)=KR+BETA*KR*VT(I)
   35 THE(I)=VT(I)+BETA*VT(I)*VT(I)/2.0
C
C     CALCULATE THE DRIVE NUMBERS
C
      DO 40 I=2,N1
      IM1=I-1
      IP1=I+1
```

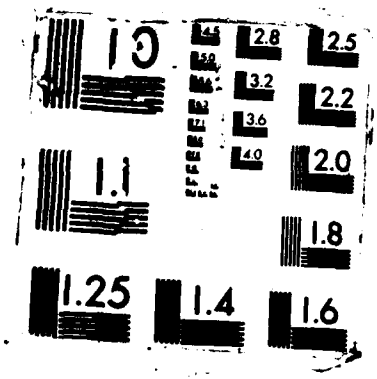
```
      M(I)=VT(IP1)+VT(IM1)-2.*VT(I)
      M(I)=M(I)+BETA*(VT(IP1)**2.+VT(IM1)**2.-2.*VT(I)**2.)/2.
 40   M(I)=M(I)/THE(I)
      DO 50 I1=2,N1
 50   KAPPA(I1)=THE(I1)*DEXP(TS*(1.+BETA*VT(I1))*M(I1))
C
C     CALCULATE THE DEPENDENT VARIABLE ON THE SUB-TIME INTERVAL
C
      DO 55 I1=2,N1
 55   VT(I1)=(-1.+SQRT(1.+2.*KAPPA(I1)*BETA))/BETA
C
C     SUM THE DRIVE NUMBERS
C
      DO 60 I=2,N1
 60   P(I)=P(I)+M(I)
 30   CONTINUE
      WRITE(6,5)
 5    FORMAT(/)
      WRITE(6,31)L
 31   FORMAT(1X,'TIME STEP NUMBER=',I3/)
      TIME=T*DFLOAT(L)
      WRITE(6,32)TIME
 32   FORMAT(5X,'ELAPSED TIME=',F10.4,'SECONDS'/)
C
C     CALCULATE THE NEXT TOTAL STEP DEPENDENT VARIABLES AND PRINT RESULTS
C
      DO 70 I=1,N
      THE(I)=V(I)+BETA*V(I)*V(I)/2.0
      KAPPA(I)=THE(I)*DEXP(TS*(1.+BETA*V(I))*P(I))
 70   V(I)=(-1.+SQRT(1.+2.*KAPPA(I)*BETA))/BETA
      WRITE(6,81)(V(I),I=1,11)
 81   FORMAT(1X,11(F8.5,2X))
 20   CONTINUE
      RETURN
      END
```

```
C
C       WRITTEN BY R.F. HANDSCHUH
C
C
C       SOURCE.EXPVAR
C
C       ***** PROGRAM #5 *****
C
C
C       THIS PROGRAM IS FOR THE SOLUTION OF THE DIFFUSION EQUATION
C       WITH VARYING THERMAL CONDUCTIVITY.  THE METHOD USED IS THE
C       EXPLICIT FINITE DIFFERENCE METHOD.
C
C
        IMPLICIT REAL*8(A-H,O-Z)
        REAL*8 V(100),KR
C
C       INPUT PROGRAM DATA
C
        WRITE(6,15)
 15     FORMAT(1X,'NUMBER OF NODES=N   I3'/)
        READ(9,10)N
 10     FORMAT(I3)
        WRITE(6,16)
 16     FORMAT(1X,'TOTAL NUMBER OF TIME STEPS= NTOT   I3')
        READ(9,21)NTOT
 21     FORMAT(I3)
        WRITE(6,24)
 24     FORMAT(1X,'INPUT TIME*THERMAL DIFFUSIVITY / LENGTH SQUARED  F5.3')
        READ(9,25)TSI
 25     FORMAT(F5.3)
        WRITE(6,22)
 22     FORMAT(1X,'TOTAL TIME OF ONE TIME STEP= T F7.5')
        READ(6,23)T
 23     FORMAT(F7.5)
        WRITE(6,14)
 14     FORMAT(1X,'INPUT REFERENCE THERMAL CONDUCTIVITY=F5.4')
        READ(9,17)KR
 17     FORMAT(F5.4)
        WRITE(6,26)
 26     FORMAT(1X,'INPUT THERMAL CONDUCTIVITY SLOPE VALUE= F5.4')
        READ(9,27)BETA
 27     FORMAT(F5.4)
C
C       INITIALIZE THE BOUNDRY CONDITIONS
C
        V(1)=0.
        V(N)=0.
        N1=N-1
        DO 30 I=2,N1
 30     V(I)=100.
C
C       CALL EXP FIN DIF FOR INFINITE HEAT TRANSFER COEFFICENT
C
```

```
      CALL EFDIHC(N,NTOT,TSI,V,T,KR,BETA)
      STOP
      END
C
C     SUBROUTINE EFDIHC
C
      SUBROUTINE EFDIHC(N,NTOT,TSI,V,T,KR,BETA)
C
C     FOR INFINITE HEAT TRANSFER COEFFICENT
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 VT(100),V(100),M(100),P(100)
      REAL*8 KR
      N1=N-1
C
C     BEGIN TIME STEP LOOP
C
      DO 20 L=1,NTOT
C
C     CALCULATE DEPENDENT VARIABLE
C
      DO 40 I=2,N1
      IM1=I-1
      IP1=I+1
      VT(I)=V(I)+TSI*((1.+BETA*V(I))*(V(IP1)+V(IM1)-2.*V(I)))
      DERSQR=DABS(V(IP1)-V(IM1))
      IF(DERSQR.LE.0.0)GO TO 40
      VT(I)=VT(I)+TSI*((BETA*(DERSQR)**2.)/4.)
 40   CONTINUE
C
C     PRINT RESULTS
C
      WRITE(6,5)
 5    FORMAT(/)
      WRITE(6,31)L
 31   FORMAT(1X,'TIME STEP NUMBER=',I3/)
      TIME=T*DFLOAT(L)
      WRITE(6,32)TIME
 32   FORMAT(5X,'ELAPSED TIME=',F10.4,'SECONDS'/)
      DO 70 I=1,N
 70   V(I)=VT(I)
      WRITE(6,81)(V(I),I=1,N)
 81   FORMAT(1X,11(F9.5,2X))
 20   CONTINUE
      RETURN
      END
```

AD-A183 901 AN EXPONENTIAL FINITE DIFFERENCE TECHNIQUE FOR SOLVING 2/2
PARTIAL DIFFERENTI..(U) GAERTNER (W W) RESEARCH INC
NORWALK CT R F HANDSCHUH JUN 87 NASA-E-3544
UNCLASSIFIED USAAVSCOM-TR-87-C-19 F/G 12/2 NL




2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.8
1.25
1.4
1.6

```
C
C
C     SOURCE.IMPVAR
C
C     WRITTEN BY R.F. HANDSCHUH
C
C     ***** PROGRAM #6 *****
C
C     THIS ROUTINE IS TO BE USED FOR COMPARISION TO EXPONENTIAL
C     FINITE DIFFERENCE ALGORITHM.  THIS ROUTINE WILL USE THE
C     IMPLICIT ROUTINE TO SOLVE FOR THE TEMPERATURE FIELD USING
C     THE TRI-DIAGONAL MATRIX ALGORITHM.
C
C
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 V(100),KR
C
C     INPUT PROGRAM DATA
C
      WRITE(6,15)
 15   FORMAT(1X,'NUMBER OF NODES=N   I3'/)
      READ(9,10)N
 10   FORMAT(I3)
      WRITE(6,16)
 16   FORMAT(1X,'TOTAL NUMBER OF TIME STEPS= NTOT   I3')
      READ(9,21)NTOT
 21   FORMAT(I3)
      WRITE(6,24)
 24   FORMAT(1X,'INPUT TIME*THERMAL DIFFUSIVITY / LENGTH SQUARED  F5.3')
      READ(9,25)TSI
 25   FORMAT(F5.3)
      WRITE(6,22)
 22   FORMAT(1X,'TOTAL TIME OF ONE TIME STEP= T F7.5')
      READ(6,23)TS
 23   FORMAT(F7.5)
      WRITE(6,14)
 14   FORMAT(1X,'INPUT REFERENCE THERMAL CONDUCTIVITY=F5.4')
      READ(9,17)KR
 17   FORMAT(F5.4)
      WRITE(6,26)
 26   FORMAT(1X,'INPUT THERMAL CONDUCTIVITY SLOPE VALUE= F5.4')
      READ(9,27)BETA
 27   FORMAT(F5.4)
C
C     INITIALIZE BOUNDRY CONDITIONS
C
      V(1)=0.
      V(N)=0.
      N1=N-1
      DO 30 I=2,N1
 30   V(I)=100.
C
C     CALL IMPLICIT ROUTINE
```

```
C
      CALL IMPL(N,NTOT,TSI,V,TS,KR,BETA)
      STOP
      END
C
C     SUBROUTINE IMPL
C
      SUBROUTINE IMPL(N,NTOT,OMEGA,T,TS,KR,BETA)
C
C     FOR INFINITE HEAT TRANSFER COEFFICENT AND VARYING THERMAL CONDUCTIVITY
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 A(101),B(101),C(101),D(101),KAP(101),GAM(101),T(101)
      REAL*8 TO(101)
      REAL*8 KR
C
C     PRINT HEADING
C
      WRITE(6,331)
 331  FORMAT(1X,'*****     SOURCE.IMPVAR     *****'/)
      RHO=1.0
      CP=1.0
      N1=N-1
C
C     BEGIN TIME STEP LOOP
C
      DO 20 L=1,NTOT
C
C     CALCULATE THOMAS ALGORITHM VARIABLES AND THOSE THAT ARE A FUNCTION
C     OF TEMPERATURE
C
      DO 21 I=1,N
 21   D(I)=T(I)
      DO 200 I=1,N
 200  TO(I)=T(I)
      DO 25 I=2,N1
      KAP(I)=1.0+BETA*T(I)
 25   GAM(I)=BETA*(T(I+1)-T(I-1))/(4.)
C
      DO 30 I=2,N1
      A(I)=-OMEGA*(KAP(I)-GAM(I))
      B(I)=(1.+2.*OMEGA*KAP(I))
 30   C(I)=-OMEGA*(KAP(I)+GAM(I))
C
C     CALL TRI-DIAGONAL-MATRIX ALGORITHM
C
      CALL TRIDAG(2,N1,A,B,C,D,T)
C
60    CONTINUE
C
C     PRINT THE RESULTS
C
      WRITE(6,5)
 5    FORMAT(/)
```

```
      WRITE(6,31)L
 31   FORMAT(1X,'TIME STEP NUMBER=',I3/)
      TIME=TS*DFLOAT(L)
      WRITE(6,32)TIME
 32   FORMAT(5X,'ELAPSED TIME=',F10.4,'SECONDS'/)
      WRITE(6,81)(T(I),I=1,N)
 81   FORMAT(1X,11(F9.5,2X))
 20   CONTINUE
      RETURN
      END
C
C     SUBROUTINE TRIDAG
C
C     THIS ROUTINE IS FOR THE SOLUTION OF THE THOMAS ALGORITHM
C
C     THIS ROUTINE WAS TAKEN FROM THE BOOK APPLIED NUMERICAL METHODS
C     BY CARNAHAN, LUTHER, AND WILKES.
C
      SUBROUTINE TRIDAG(IF,L,A,B,C,D,V)
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 A(101),B(101),C(101),D(101),V(101),BETA(101),GAMMA(101)
C
C     COMPUTE INTERMEDIATE ARRAYS BETA AND GAMMA
C
      BETA(IF)=B(IF)
      GAMMA(IF)=D(IF)/BETA(IF)
      IFP1=IF+1
      DO 1 I=IFP1,L
      BETA(I)=B(I)-A(I)*C(I-1)/BETA(I-1)
  1   GAMMA(I)=(D(I)-A(I)*GAMMA(I-1))/BETA(I)
C
C     COMPUTE FINAL SOLUTION VECTOR V
C
      V(L)=GAMMA(L)
      LAST=L-IF
      DO 2 K=1,LAST
      I=L-K
 2    V(I)=GAMMA(I)-C(I)*V(I+1)/BETA(I)
      RETURN
      END
```

```
C
C
C     SOURCE.COUE
C
C     WRITTEN BY R.F. HANDSCHUH
C
C     ***** PROGRAM #7 *****
C
C     THIS PROGRAM IS TO BE USED TO DEMONSTRATE THE USE OF THE
C     EXPONENTIAL FINITE DIFFERENCE METHOD ON THE DEVELOPING TEMPERATURE
C     FIELD IN A LAMINAR COUETTE FLOW
C
C
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 V(100)
C
C     INPUT PROGRAM DATA
C
      WRITE(6,15)
 15   FORMAT(1X,'NUMBER OF NODES=N   I3'/)
      READ(9,10)N
 10   FORMAT(I3)
      WRITE(6,12)
 12   FORMAT(1X,'NUMBER OF SUB INTERVALS= NS  I3')
      READ(6,13)NS
 13   FORMAT(I3)
      WRITE(6,16)
 16   FORMAT(1X,'TOTAL NUMBER OF POSITION STEPS= NTOT   I3')
      READ(9,21)NTOT
 21   FORMAT(I3)
      WRITE(6,24)
 24   FORMAT(1X,'INPUT (POSITION STEP*KIN. VISC)/(LENGTH SQUARED) F5.3')
      READ(9,25)TSI
 25   FORMAT(F5.3)
      WRITE(6,22)
 22   FORMAT(1X,'TOTAL DISTANCE OF ONE STEP= T F5.3')
      READ(6,23)T
 23   FORMAT(F5.3)
      WRITE(6,14)
 14   FORMAT(1X,'INPUT (MAX WIDTH)/(FLOW VEL)  F5.4')
      READ(6,17)SP
 17   FORMAT(F5.4)
      WRITE(6,31)
 31   FORMAT(1X,'INPUT INTERVAL FOR PRINTING RESULTS= I3')
      READ(6,32)IPR
 32   FORMAT(I3)
C
C
C     INITIALIZE THE BOUNDRY CONDITIONS
C
      V(1)=0.
      DO 30 I=2,N
 30   V(I)=100.
```

```
C
C     CALL EXPONENTIAL FINITE DIFFERENCE FOR COUETTE FLOW
C
      CALL EFFL(N,NS,NTOT,TSI,V,T,SP,IPR)
      STOP
      END
C
C     SUBROUTINE EFFL
C
      SUBROUTINE EFFL(N,NS,NTOT,TSI,V,T,SP,IPR)
C
C     FOR COUETTE LAMINAR FLOW
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 VT(100),V(100),M(100),P(100),FL(20)
      TS=TSI/DFLOAT(NS+1)
      N1=N-1
C
C     PRINT HEADING
C
      WRITE(6,92)
 92   FORMAT(1X,'SOLUTION FOR DEVELOPING TEMPERATURE FIELD IN'        -
     *'LAMINAR COUETTE FLOW'//)
      DY=1./DFLOAT(N-1)
C
C     CALCULATE PARAMETER THAT VARIES WITH POSITION IN THE FLOW
C
      DO 115 I=2,N
 115  FL(I)=SP/(DY*DFLOAT(I-1))
      NS1=NS+1
C
C     BEGIN TOTAL POSITION STEP LOOP
C
      DO 20 L=1,NTOT
      IT=IT+1
C
C     ZERO THE SUM OF DRIVE NUMBERS
C
      DO 15 I=1,N
 15   P(I)=0.
C
C     SET TEMPORARY VALUES EQUAL TO THE LAST POSITION STEP VALUE
C
      DO 10 I=1,N
 10   VT(I)=V(I)
C
C     SUB POSITION INTERVAL
C
      DO 30 K=1,NS1
C
C     CALCULATE THE DRIVE NUMBERS
C
      DO 40 I=2,N1
      IM1=I-1
```

```
      IP1=I+1
 40   M(I)=(2.*VT(I)-VT(IM1)-VT(IP1))/VT(I)
C
C     CALCULATE TEMPORARY DEPENDENT VARIABLE ON SUB-INTERVAL
C
      DO 50 I1=2,N1
 50   VT(I1)=VT(I1)*DEXP(-TS*M(I1)*FL(I1))
      DO 60 I=2,N1
 60   P(I)=P(I)+M(I)
 30   CONTINUE
C
C     CALCULATE THE COMPLETE STEP DEPENDENT VARIABLES
C
      DO 70 I=1,N
      V(I)=V(I)*DEXP(-TS*P(I)*FL(I))
 70   CONTINUE
      IF(IT.LT.IPR)GO TO 20
      IT=0
C
C     PRINT THE RESULTS
C
      WRITE(6,5)
 5    FORMAT(/)
      WRITE(6,31)L
 31   FORMAT(1X,'POSITON STEP NUMBER=',I3)
      DIST=T*DFLOAT(L)
      WRITE(6,32)DIST
 32   FORMAT(5X,'LOCATION DIST=',F10.4,'METERS')
      WRITE(6,82)(V(I),I=1,N)
 82   FORMAT(1X,11(F9.5,2X))
 20   CONTINUE
      RETURN
      END
```

```
C
C
C     WRITTEN BY R. F. HANDSCHUH
C
C     SOURCE.EX3D
C
C     ***** PROGRAM #8 *****
C
C
C     THIS ROUTINE IS FOR FINDING THE TEMPERATURE AT A GIVEN LOCATION
C     AND TIME FOR A 3-DIMENSIONAL SOLID.
C
C     THIS PROGRAM CALCULATES THE EXACT TEMP AS FOUND IN THE BOOK
C     "TRANSPORT PHENOMENA" BY BIRD, STEWART, AND LIGHTFOOT.
C
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 PI,PI2,PI3
      PI=3.1415926
C
C     INPUT PROGRAM DATA
C
      WRITE(1,31)
 31   FORMAT(1X,'NUMBER OF NODES PER COORDINATE DIRECTION=I3')
      READ(5,32)NODES
 32   FORMAT(I3)
      WRITE(1,1)
 1    FORMAT(1X,'INPUT INITIAL TEMPERATURE FOR THE SOLID  F6.3')
      READ(5,2)T0
 2    FORMAT(F6.3)
      WRITE(1,3)
 3    FORMAT(1X,'SURFACE TEMPERATURE FOR TIME > 0 SECONDS   F6.3')
      READ(5,4)T1
 4    FORMAT(F6.3)
      WRITE(1,5)
 5    FORMAT(1X,'INPUT THERMAL DIFFUSIVITY F5.3')
      READ(5,6)ALFA
 6    FORMAT(F5.3)
C
C     ASSUMING 3 DIM. CUBE
C
      DATA A,B,C/1.0,1.0,1.0/
      WRITE(1,9)
 9    FORMAT(1X,'INPUT THE NUMBER OF TIMES THROUGH SUMMATION I3')
      READ(5,11)N1
 11   FORMAT(I3)
      WRITE(1,12)
 12   FORMAT(1X,'INPUT TIME TO BE EVALUATED AT F5.3')
      READ(5,14)TIME
 14   FORMAT(F5.3)
      WRITE(6,20)T0,T1
 20   FORMAT(1X,'TEMP INITIAL=',E15.8,2X,'TEMP SURF=',E15.8/)
      WRITE(6,30)ALFA,TIME
 30   FORMAT(1X,'THER DIFF=',E15.8,2X,'TIME=',F7.5)
```

```
C
C       START SUMMATION ROUTINE
C
        PI3=PI**3.
        PI2=PI**2.
C
C       CALCULATE THE TEMPERATURE ALONG THE DIAGONAL
C
        DEL=A/(DFLOAT(NODES-1))
C
        DO 100 NODE=1,NODES
        X=DEL*DFLOAT(NODE-1)
        Y=X
        Z=X
        SUM=0.0
        DO 10 K=1,N1
        RP=DFLOAT(K-1)
        RP1=RP+.5
        COSP=DCOS(RP1*PI*Z/C)
        DO 10 N=1,N1
        RN=DFLOAT(N-1)
        RN1=RN+.5
        COSN=DCOS(RN1*PI*Y/B)
        DO 10 M=1,N1
        RM=DFLOAT(M-1)
        RM1=RM+.5
        COSM=DCOS(RM1*PI*X/A)
        J=M+N+K-3
        J1=J/2
        J2=J1*2
        VAL=-1.0
        IF(J.EQ.J2)VAL=1.0
        GAM=(RM1**2)/(A*A)+(RN1**2)/(B*B)+(RP1**2)/(C*C)
        EXPLIM=(-GAM*PI2*ALFA*TIME)
        IF(EXPLIM.LT.-100.)GO TO 999
        EP=DEXP(EXPLIM)
        GO TO 998
 999    CONTINUE
        EP=0.0
 998    CONTINUE
        SUM=SUM+(VAL/(RM1*RN1*RP1*PI3))*EP*COSM*COSN*COSP
 10     CONTINUE
        TNEW=T1+8.*(T0-T1)*SUM
C
C       PRINT THE RESULTS
C
        WRITE(6,16)X,Y,Z
 16     FORMAT(1X,'X=',F5.3,2X,'Y=',F5.3,2X,'Z=',F5.3)
        IF(TNEW.LT.1E-10)TNEW=0.0
        WRITE(6,15)TNEW
 15     FORMAT(1X,'TEMPERATURE=',E15.8)
 100    CONTINUE
        STOP
        END
```

```
C
C     SOURCE.EFD3D
C
C     WRITTEN BY R.F. HANDSCHUH
C
C     ***** PROGRAM #9 *****
C
C     THIS PROGRAM IS FOR 3-DIMENSIONAL CARTESIAN COORDINATES
C     UNSTEADY STATE HEAT TRANSFER IN A CUBE.  THE METHOD OF SOLUTION IS THE
C     EXPONENTIAL FINITE DIFFERENCE ALGORYTHM.  THIS PARTICULAR PROGRAM IS
C     FOR INFINITE HEAT TRANSFER COEFFICENT AT THE EXPOSED SURFACES
C     AT X=Y=Z=1.0  FOR THE THREE SURFACES WHERE X,Y,Z EQUAL 0.0
C     ARE TO BE CONSIDERED AS PERFECTLY INSULATED.
C
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 V(25,25,25)
C
C     INPUT PROGRAM DATA
C
      WRITE(6,15)
 15   FORMAT(1X,'NUMBER OF NODES=N   I3'/)
      READ(9,10)N
 10   FORMAT(I3)
      WRITE(6,12)
 12   FORMAT(1X,'NUMBER OF TIME SUB INTERVALS= NS  I3')
      READ(9,13)NS
 13   FORMAT(I3)
      WRITE(6,16)
 16   FORMAT(1X,'TOTAL NUMBER OF TIME STEPS= NTOT   I3')
      READ(9,21)NTOT
 21   FORMAT(I3)
      WRITE(6,24)
 24   FORMAT(1X,'INPUT TIME*THERMAL DIFFUSIVITY / LENGTH SQUARED  F5.3')
      READ(9,25)TSI
 25   FORMAT(F5.3)
      WRITE(6,22)
 22   FORMAT(1X,'TOTAL TIME OF ONE TIME STEP= T F6.4')
      READ(9,23)T
 23   FORMAT(F6.4)
      WRITE(6,26)
 26   FORMAT(1X,'NUMBER OF TIME STEPS BEFORE PRINTING= I3')
      READ(9,27)IPR
 27   FORMAT(I3)
      WRITE(6,250)N,NS,NTOT
 250  FORMAT(1X,'# OF NODES=',I3,2X,'# OF SUB-TIME-INT=',I3,2X,       -
     *'# OF TIME STEPS=',I3)
      WRITE(6,251)TSI,T
 251  FORMAT(1X,'(TIME*THER DIFF)/LENGTH SQUARED='F5.3,2X,            -
     *'TIME STEP LENGTH=',F6.4/)
      N1=N-1
C
C     INITIALIZE BOUNDRY CONDITIONS
C
```

```
      DO 30 I=1,N1
      DO 30 J=1,N1
      DO 30 K=1,N1
 30   V(I,J,K)=1.0
C
      I=N
      DO 50 J=1,N
      DO 50 K=1,N
 50   V(I,J,K)=0.0
      J=N
      DO 51 I=1,N
      DO 51 K=1,N
 51   V(I,J,K)=0.0
      K=N
      DO 52 I=1,N
      DO 52 J=1,N
 52   V(I,J,K)=0.0
C
C     CALL EXP FIN DIF FOR INFINITE HEAT TRANSFER COEFFICENT
C
      CALL EFDIHC(N,NS,NTOT,TSI,V,T,IPR)
      STOP
      END
C
C     SUBROUTINE EFDIHC
C
      SUBROUTINE EFDIHC(N,NS,NTOT,TSI,V,T,IPR)
C
C     FOR INFINITE HEAT TRANSFER COEFFICENT
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 VT(25,25,25),V(25,25,25),M(25,25,25),P(25,25,25)
      REAL*8 M1,M2
      TS=TSI/DFLOAT(NS+1)
      N1=N-1
      NS1=NS+1
C
C     PRINT HEADING
C
      WRITE(6,200)
 200  FORMAT(1X,'********** RESULTS FROM EFD3D **********'//)
C
C     START TOTAL TIME STEP LOOP
C
      DO 20 L=1,NTOT
C
C     ZERO THE SUM OF THE SUB-INTERVAL DRIVE NUMBERS
C
      DO 15 K=1,N
      DO 15 J=1,N
      DO 15 I=1,N
 15   P(I,J,K)=0.
C
C     SET SUB-INTERVAL VALUES EQUAL TO THE LAST TIME STEP VALUES
```

```
C
      DO 10 K=1,N
      DO 10 J=1,N
      DO 10 I=1,N
 10   VT(I,J,K)=V(I,J,K)
C
C     SUB TIME INTERVAL
C
C     CALCULATE THE DRIVE NUMBERS WHICH IS DEPENDENT ON LOCATION IN THE CUBE
C
      DO 30 KS=1,NS1
      DO 42 K=2,N1
      KM1=K-1
      KP1=K+1
      DO 41 J=2,N1
      JM1=J-1
      JP1=J+1
      DO 40 I=2,N1
      IM1=I-1
      IP1=I+1
      M1=VT(IP1,J,K)+VT(IM1,J,K)+VT(I,JP1,K)+VT(I,JM1,K)
      M2=M1+VT(I,J,KP1)+VT(I,J,KM1)-6.*VT(I,J,K)
      IF(VT(I,J,K).LE.0.0)M(I,J,K)=0.
      IF(VT(I,J,K).LE.0.0)GO TO 40
      M(I,J,K)=M2/VT(I,J,K)
 40   CONTINUE
 41   CONTINUE
 42   CONTINUE
C
C     INSULATED BOUNDRY ALONG X-AXIS
C
      J=1
      K=1
      KP1=K+1
      JP1=J+1
      DO 48 I=2,N1
      IP1=I+1
      IM1=I-1
      M1=VT(IP1,J,K)+VT(IM1,J,K)+2.*VT(I,JP1,K)+2.*VT(I,J,KP1)
      IF(VT(I,J,K).LE.0.0)M(I,J,K)=0.
      IF(VT(I,J,K).LE.0.0)GO TO 48
      M(I,J,K)=(M1-6.*VT(I,J,K))/VT(I,J,K)
 48   CONTINUE
C
C     INSULATED BOUNDRY ALONG Y-AXIS
C
      I=1
      IP1=I+1
      K=1
      KP1=K+1
      DO 43 J=2,N1
      JP1=J+1
      JM1=J-1
      M1=2.*VT(IP1,J,K)+VT(I,JP1,K)+VT(I,JM1,K)+2.*VT(I,J,KP1)
```

```
      IF(VT(I,J,K).LE.0.0)M(I,J,K)=0.
      IF(VT(I,J,K).LE.0.0)GO TO 43
      M(I,J,K)=(M1-6.*VT(I,J,K))/VT(I,J,K)
 43   CONTINUE
C
C     INSULATED BOUNDRY ALONG Z-AXIS
C
      J=1
      JP1=J+1
      I=1
      IP1=I+1
      DO 44 K=2,N1
      KM1=K-1
      KP1=K+1
      M1=2.*VT(IP1,J,K)+2.*VT(I,JP1,K)+VT(I,J,KP1)+VT(I,J,KM1)
      IF(VT(I,J,K).LE.0.0)M(I,J,K)=0.
      IF(VT(I,J,K).LE.0.0)GO TO 44
      M(I,J,K)=(M1-6.*VT(I,J,K))/VT(I,J,K)
 44   CONTINUE
C
C     INSULATED FACE AT Z=0
C
      K=1
      KP1=K+1
      DO 45 I=2,N1
      IP1=I+1
      IM1=I-1
      DO 45 J=2,N1
      JP1=J+1
      JM1=J-1
      M1=VT(IP1,J,K)+VT(IM1,J,K)+VT(I,JP1,K)+VT(I,JM1,K)+2.*VT(I,J,KP1)
      IF(VT(I,J,K).LE.0.0)M(I,J,K)=0.
      IF(VT(I,J,K).LE.0.0)GO TO 45
      M(I,J,K)=(M1-6.*VT(I,J,K))/VT(I,J,K)
 45   CONTINUE
C
C     AT THE FACE WHERE Y=0
C
      J=1
      JP1=J+1
      DO 46 I=2,N1
      IP1=I+1
      IM1=I-1
      DO 46 K=2,N1
      KP1=K+1
      KM1=K-1
      M1=VT(IP1,J,K)+VT(IM1,J,K)+2.*VT(I,JP1,K)+VT(I,J,KP1)+VT(I,J,KM1)
      IF(VT(I,J,K).LE.0.0)M(I,J,K)=0.
      IF(VT(I,J,K).LE.0.0)GO TO 46
      M(I,J,K)=(M1-6.*VT(I,J,K))/VT(I,J,K)
 46   CONTINUE
C
C     AT THE FACE WHERE X=0
C
```

```
      I=1
      IP1=I+1
      DO 47 J=2,N1
      JP1=J+1
      JM1=J-1
      DO 47 K=2,N1
      KP1=K+1
      KM1=K-1
      M1=2.*VT(IP1,J,K)+VT(I,JP1,K)+VT(I,JM1,K)+VT(I,J,KP1)+VT(I,J,KM1)
      IF(VT(I,J,K).LE.0.0)M(I,J,K)=0.
      IF(VT(I,J,K).LE.0.0)GO TO 47
      M(I,J,K)=(M1-6.*VT(I,J,K))/VT(I,J,K)
 47   CONTINUE
C
C     CORNER AT ORIGIN
C
      M1=2.*VT(1,2,1)+2.*VT(2,1,1)+2.*VT(1,1,2)-6.*VT(1,1,1)
      M(1,1,1)=M1/VT(1,1,1)
C
C
C
C     CALCULATE THE SUB-INTERVAL DEPENDENT VARIABLES
C
      DO 50 I1=1,N
      DO 50 J1=1,N
      DO 50 K1=1,N
      IF(M(I1,J1,K1).LT.-50.)VT(I1,J1,K1)=0.0
      IF(M(I1,J1,K1).LT.-50)GO TO 50
      VT(I1,J1,K1)=VT(I1,J1,K1)*DEXP(TS*M(I1,J1,K1))
 50   CONTINUE
C
C     SUM THE DRIVE NUMBERS
C
      DO 60 I=1,N
      DO 60 J=1,N
      DO 60 K=1,N
 60   P(I,J,K)=P(I,J,K)+M(I,J,K)
 30   CONTINUE
C
C     CALCULATE THE NEXT COMPLETE TIME STEP DEPENDENT VARIABLES
C
      DO 70 K=1,N
      DO 70 J=1,N
      DO 70 I=1,N
      IF(P(I,J,K).LT.-50.)V(I,J,K)=0.0
      IF(P(I,J,K).LT.-50.)GO TO 70
      V(I,J,K)=V(I,J,K)*DEXP(TS*P(I,J,K))
 70   CONTINUE
      ITMAX=ITMAX+1
      IF(ITMAX.LT.IPR)GO TO 20
      WRITE(6,5)
 5    FORMAT(/)
      WRITE(6,31)L
 31   FORMAT(1X,'TIME STEP NUMBER=',I3/)
```

```
      TIME=T*DFLOAT(L)
      WRITE(6,32)TIME
 32   FORMAT(5X,'ELAPSED TIME=',F10.4,'SECONDS'/)
C
C     PRINT OUT THE DIAGONAL RESULTS
C
      WRITE(6,82)(V(I,I,I),I=1,N)
 82   FORMAT(11(2X,F8.6))
      ITMAX=0
 20   CONTINUE
      RETURN
      END
```

```
C
C     SOURCE.EXPL3D
C
C     WRITTEN BY R.F. HANDSCHUH
C
C     ***** PROGRAM #10 *****
C
C     THIS PROGRAM IS FOR 3-DIMENSIONAL CARTESIAN COORDINATES
C     UNSTEADY STATE HEAT TRANSFER IN A CUBE.  THE METHOD OF SOLUTION IS THE
C     PURE EXPLICIT METHOD.  THIS PARTICULAR PROGRAM IS
C     FOR INFINITE HEAT TRANSFER COEFFICENT AT THE EXPOSED SURFACES
C     AT X=Y=Z=1.0  FOR THE THREE SURFACES WHERE X,Y,Z EQUAL 0.0
C     IS TO BE CONSIDERED AS PERFECTLY INSULATED.
C
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 V(25,25,25)
C
C     INPUT PROGRAM DATA
C
C     NUMBER OF NODES = N
C
      READ(5,10)N
 10   FORMAT(I3)
C
C     TOTAL NUMBER OF TIME STEPS = NTOT
C
      READ(5,21)NTOT
 21   FORMAT(I4)
C
C     TIME*THERMAL DIFFUSIVITY/LENGTH SQUARED = TSI
C
      READ(5,25)TSI
 25   FORMAT(F5.3)
C
C     TOTAL TIME OF ONE TIME STEP = T
C
      READ(5,23)T
 23   FORMAT(F6.4)
C
C     NUMBER OF STEPS BEFORE PRINTING THE RESULTS = IPR
C
      READ(5,27)IPR
 27   FORMAT(I3)
      WRITE(6,252)
 252  FORMAT(1X,'*******************  PURE EXLICIT FINITE DIFFERENCE  ' -
     *'METHOD  ****************'///)
      WRITE(6,250)N,NS,NTOT
 250  FORMAT(1X,'# OF NODES=',I3,2X,'# OF SUB-TIME-INT=',I3,2X,       -
     *'# OF TIME STEPS=',I4)
      WRITE(6,251)TSI,T
 251  FORMAT(1X,'(TIME*THER DIFF)/LENGTH SQUARED='F5.3,2X,            -
     *'TIME STEP LENGTH=',F6.4/)
      N1=N-1
```

```
C
C     INITIALIZE BOUNDRY CONDITIONS
C
      DO 30 I=1,N1
      DO 30 J=1,N1
      DO 30 K=1,N1
 30   V(I,J,K)=1.0
C
      I=N
      DO 50 J=1,N
      DO 50 K=1,N
 50   V(I,J,K)=0.0
      J=N
      DO 51 I=1,N
      DO 51 K=1,N
 51   V(I,J,K)=0.0
      K=N
      DO 52 I=1,N
      DO 52 J=1,N
 52   V(I,J,K)=0.0
C
C     CALL PURE EXPLICIT FINITE DIFFERENCE FOR INFINITE HEAT TRANSFER COEFFICENT
C
      CALL PURE(N,NTOT,TSI,V,T,IPR)
      STOP
      END
C
C     SUBROUTINE PURE
C
      SUBROUTINE PURE(N,NTOT,TSI,V,T,IPR)
C
C     PURE EXPLICIT FINITE DIFFERENCE METHOD
C     FOR INFINITE HEAT TRANSFER COEFFICENT
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 V(25,25,25),VT(25,25,25)
      REAL*8 M1,M2
      N1=N-1
C
C     START TIME STEP LOOP
C
      DO 20 L=1,NTOT
C
C     SAVE VALUES FROM THE LAST TIME STEP
C
      DO 39 I=1,N
      DO 39 J=1,N
      DO 39 K=1,N
 39   VT(I,J,K)=V(I,J,K)
C
C     CALCULATE THE FIELD VARIABLE USING THE EXPLICIT FINITE DIFFERENCE METHOD
C
      DO 42 K=2,N1
      KM1=K-1
```

```
      KP1=K+1
      DO 41 J=2,N1
      JM1=J-1
      JP1=J+1
      DO 40 I=2,N1
      IM1=I-1
      IP1=I+1
      M1=VT(IP1,J,K)+VT(IM1,J,K)+VT(I,JP1,K)+VT(I,JM1,K)
      M2=M1+VT(I,J,KP1)+VT(I,J,KM1)-6.*VT(I,J,K)
      V(I,J,K)=V(I,J,K)+TSI*M2
 40   CONTINUE
 41   CONTINUE
 42   CONTINUE
C
C     INSULATED BOUNDRY ALONG X-AXIS
C
      J=1
      K=1
      KP1=K+1
      JP1=J+1
      DO 48 I=2,N1
      IP1=I+1
      IM1=I-1
      M1=VT(IP1,J,K)+VT(IM1,J,K)+2.*VT(I,JP1,K)+2.*VT(I,J,KP1)
      M2=M1-6.*VT(I,J,K)
      V(I,J,K)=V(I,J,K)+TSI*M2
 48   CONTINUE
C
C     INSULATED BOUNDRY ALONG Y-AXIS
C
      I=1
      IP1=I+1
      K=1
      KP1=K+1
      DO 43 J=2,N1
      JP1=J+1
      JM1=J-1
      M1=2.*VT(IP1,J,K)+VT(I,JP1,K)+VT(I,JM1,K)+2.*VT(I,J,KP1)
      M2=M1-6.*VT(I,J,K)
      V(I,J,K)=V(I,J,K)+TSI*M2
 43   CONTINUE
C
C     INSULATED BOUNDRY ALONG Z-AXIS
C
      J=1
      JP1=J+1
      I=1
      IP1=I+1
      DO 44 K=2,N1
      KM1=K-1
      KP1=K+1
      M1=2.*VT(IP1,J,K)+2.*VT(I,JP1,K)+VT(I,J,KP1)+VT(I,J,KM1)
      M2=M1-6.*VT(I,J,K)
      V(I,J,K)=V(I,J,K)+TSI*M2
```

```
 44   CONTINUE
C
C     INSULATED FACE AT Z=0
C
      K=1
      KP1=K+1
      DO 45 I=2,N1
      IP1=I+1
      IM1=I-1
      DO 45 J=2,N1
      JP1=J+1
      JM1=J-1
      M1=VT(IP1,J,K)+VT(IM1,J,K)+VT(I,JP1,K)+VT(I,JM1,K)+2.*VT(I,J,KP1)
      M2=M1-6.*VT(I,J,K)
      V(I,J,K)=V(I,J,K)+TSI*M2
 45   CONTINUE
C
C     AT THE FACE WHERE Y=0
C
      J=1
      JP1=J+1
      DO 46 I=2,N1
      IP1=I+1
      IM1=I-1
      DO 46 K=2,N1
      KP1=K+1
      KM1=K-1
      M1=VT(IP1,J,K)+VT(IM1,J,K)+2.*VT(I,JP1,K)+VT(I,J,KP1)+VT(I,J,KM1)
      M2=M1-6.*VT(I,J,K)
      V(I,J,K)=V(I,J,K)+TSI*M2
 46   CONTINUE
C
C     AT THE FACE WHERE X=0
C
      I=1
      IP1=I+1
      DO 47 J=2,N1
      JP1=J+1
      JM1=J-1
      DO 47 K=2,N1
      KP1=K+1
      KM1=K-1
      M1=2.*VT(IP1,J,K)+VT(I,JP1,K)+VT(I,JM1,K)+VT(I,J,KP1)+VT(I,J,KM1)
      M2=M1-6.*VT(I,J,K)
      V(I,J,K)=V(I,J,K)+TSI*M2
 47   CONTINUE
C
C     CORNER AT ORIGIN
C
      M1=2.*VT(1,2,1)+2.*VT(2,1,1)+2.*VT(1,1,2)-6.*VT(1,1,1)
      V(1,1,1)=V(1,1,1)+M1*TSI
C
C
      ITMAX=ITMAX+1
```

```
      IF(ITMAX.LT.IPR)GO TO 20
      WRITE(6,5)
 5    FORMAT(/)
      WRITE(6,31)L
 31   FORMAT(1X,'TIME STEP NUMBER=',I3/)
      TIME=T*DFLOAT(L)
      WRITE(6,32)TIME
 32   FORMAT(5X,'ELAPSED TIME=',F10.4,'SECONDS'/)
C
C     PRINT OUT THE DIAGONAL RESULTS
C
      WRITE(6,82)(V(I,I,I),I=1,N)
 82   FORMAT(11(2X,F8.6))
      ITMAX=0
 20   CONTINUE
      RETURN
      END
```

```
C
C     SOURCE.DOUGLA
C
C
C     WRITTEN BY R.F. HANDSCHUH
C
C     ***** PROGRAM #11 *****
C
C     THIS PROGRAM IS FOR 3-DIMENSIONAL CARTESIAN COORDINATES
C     UNSTEADY STATE HEAT TRANSFER IN A CUBE.  THE METHOD OF SOLUTION IS THE
C     METHOD OF DOUGLAS.  THIS PARTICULAR PROGRAM IS
C     FOR INFINITE HEAT TRANSFER COEFFICENT AT THE EXPOSED SURFACES
C     AT X=Y=Z=1.0  FOR THE THREE SURFACES WHERE X,Y,Z EQUAL 0.0
C     IS TO BE CONSIDERED AS PERFECTLY INSULATED.
C
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 T(25,25,25)
C
C     INPUT PROGRAM DATA
C
      WRITE(6,15)
 15   FORMAT(1X,'NUMBER OF NODES=N   I3'/)
      READ(5,10)N
 10   FORMAT(I3)
      WRITE(6,16)
 16   FORMAT(1X,'TOTAL NUMBER OF TIME STEPS= NTOT   I3')
      READ(5,21)NTOT
 21   FORMAT(I3)
      WRITE(6,24)
 24   FORMAT(1X,'INPUT TIME*THERMAL DIFFUSIVITY / LENGTH SQUARED  F5.3')
      READ(5,25)TSI
 25   FORMAT(F5.3)
      WRITE(6,22)
 22   FORMAT(1X,'TOTAL TIME OF ONE TIME STEP= T F6.4')
      READ(5,23)DT
 23   FORMAT(F6.4)
      WRITE(6,26)
 26   FORMAT(1X,'NUMBER OF TIME STEPS BEFORE PRINTING= I3')
      READ(5,27)IPR
 27   FORMAT(I3)
      WRITE(6,250)N,NS,NTOT
 250  FORMAT(1X,'# OF NODES=',I3,2X,'# OF SUB-TIME-INT=',I3,2X,      -
     *'# OF TIME STEPS=',I3)
      WRITE(6,251)TSI,DT
 251  FORMAT(1X,'(TIME*THER DIFF)/LENGTH SQUARED='F5.3,2X,           -
     *'TIME STEP LENGTH=',F6.4/)
      N1=N-1
C
C     INITIALIZE BOUNDRY CONDITIONS
C
      DO 30 I=1,N1
      DO 30 J=1,N1
      DO 30 K=1,N1
```

```
 30   T(I,J,K)=1.0
C
      I=N
      DO 50 J=1,N
      DO 50 K=1,N
 50   T(I,J,K)=0.0
      J=N
      DO 51 I=1,N
      DO 51 K=1,N
 51   T(I,J,K)=0.0
      K=N
      DO 52 I=1,N
      DO 52 J=1,N
 52   T(I,J,K)=0.0
C
      CALL DG(N,NS,NTOT,TSI,T,DT,IPR)
      STOP
      END
C
C     SUBROUTINE DG
C
      SUBROUTINE DG(N,NS,NTOT,TSI,T,DT,IPR)
C
C     FOR INFINITE HEAT TRANSFER COEFFICENT
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 U(25,25,25),V(25,25,25),M(25,25,25),T(25,25,25)
      REAL*8 TEMP(25),A(25),B(25),C(25),D(25)
      REAL*8 M1,M2
      N1=N-1
C
C     PRINT OUT HEADING
C
      WRITE(6,200)
 200  FORMAT(1X,'********** RESULTS FROM METHOD OF DOUGLAS **********' -
     &//)
C
C     CALCULATE COEFFICENTS FOR THOMAS ALGORITHM ( TRI-DIAGONAL MATRIX SOLVER)
C
      A(1)=0.0
      B(1)=1.+TSI
      C(1)=-TSI
      DO 60 I=2,N
      A(I)=-.5*TSI
      B(I)=1.+TSI
 60   C(I)=A(I)
C
C     BEGIN TIME STEP LOOP
C
      DO 20 L=1,NTOT
C
C     CALCULATE TEMPORARY VARIABLES "U" - X DIRECTION, ""V" - Y DIRECTION,
C     THEN ACTUAL FIELD VARIABLE "T" - Z DIRECTION.
C
```

```
      DO 120 LOOP=1,3
      DO 42 K=2,N1
      DO 41 J=2,N1
      DO 40 I=2,N1
      IF(LOOP.EQ.2)GO TO 140
      IF(LOOP.EQ.3)GO TO 141
      DELX=T(I+1,J,K)+T(I-1,J,K)-2.*T(I,J,K)
      DELY=T(I,J+1,K)+T(I,J-1,K)-2.*T(I,J,K)
      DELZ=T(I,J,K+1)+T(I,J,K-1)-2.*T(I,J,K)
      M(I,J,K)=T(I,J,K)+TSI*(.5*DELX+DELY+DELZ)
      GO TO 40
 140  CONTINUE
      DELU=U(I+1,J,K)+U(I-1,J,K)-2.*U(I,J,K)
      DELY=T(I,J+1,K)+T(I,J-1,K)-2.*T(I,J,K)
      M(I,J,K)=M(I,J,K)+.5*TSI*(DELU-DELY)
      GO TO 40
 141  CONTINUE
      DELV=V(I,J+1,K)+V(I,J-1,K)-2.*V(I,J,K)
      DELZ=T(I,J,K+1)+T(I,J,K-1)-2.*T(I,J,K)
      M(I,J,K)=M(I,J,K)+.5*TSI*(DELV-DELZ)
 40   CONTINUE
 41   CONTINUE
 42   CONTINUE
C
C     INSULATED BOUNDRY ALONG X-AXIS
C
      J=1
      K=1
      KP1=K+1
      JP1=J+1
      DO 48 I=2,N1
      IP1=I+1
      IM1=I-1
      IF(LOOP.EQ.2)GO TO 148
      IF(LOOP.EQ.3)GO TO 248
      M1=.5*(T(I+1,J,K)+T(I-1,J,K)-2.*T(I,J,K))
      M2=2.*T(I,JP1,K)+2.*T(I,J,KP1)-4.*T(I,J,K)
      M(I,J,K)=T(I,J,K)+TSI*(M1+M2)
      GO TO 48
 148  CONTINUE
      DELU=U(I+1,J,K)+U(I-1,J,K)-2.*U(I,J,K)
      DELY=2.*(T(I,J+1,K)-T(I,J,K))
      M(I,J,K)=M(I,J,K)+.5*TSI*(DELU-DELY)
      GO TO 48
 248  CONTINUE
      DELV=2.*V(I,J+1,K)-2.*V(I,J,K)
      M(I,J,K)=M(I,J,K)+.5*TSI*(DELV-2.*(T(I,J,KP1)-T(I,J,K)))
 48   CONTINUE
C
C     INSULATED BOUNDRY ALONG Y-AXIS
C
      I=1
      IP1=I+1
      K=1
```

```
      KP1=K+1
      DO 43 J=2,N1
      JP1=J+1
      JM1=J-1
      IF(LOOP.EQ.2)GO TO 143
      IF(LOOP.EQ.3)GO TO 243
      M1=T(IP1,J,K)+T(I,JP1,K)+T(I,JM1,K)+2.*T(I,J,KP1)-5.*T(I,J,K)
      M(I,J,K)=T(I,J,K)+TSI*M1
      GO TO 43
  143 CONTINUE
      DELU=2.*(U(IP1,J,K)-U(I,J,K))
      DELY=T(I,JP1,K)+T(I,JM1,K)-2.*T(I,J,K)
      M(I,J,K)=M(I,J,K)+.5*TSI*(DELU-DELY)
      GO TO 43
  243 CONTINUE
      DELV=V(I,J+1,K)+V(I,J-1,K)-2.*V(I,J,K)
      M(I,J,K)=M(I,J,K)+.5*TSI*(DELV-2.*(T(I,J,KP1)-T(I,J,K)))
   43 CONTINUE
C
C     INSULATED BOUNDRY ALONG Z-AXIS
C
      J=1
      JP1=J+1
      I=1
      IP1=I+1
      DO 44 K=2,N1
      KM1=K-1
      KP1=K+1
      IF(LOOP.EQ.2)GO TO 144
      IF(LOOP.EQ.3)GO TO 244
      M1=T(IP1,J,K)+2.*T(I,JP1,K)+T(I,J,KP1)+T(I,J,KM1)-5.*T(I,J,K)
      M(I,J,K)=T(I,J,K)+TSI*M1
      GO TO 44
  144 CONTINUE
      M1=M(I,J,K)
      M2=M1+.5*TSI*(2.*(U(IP1,J,K)-U(I,J,K))-2.*(T(I,JP1,K)-T(I,J,K)))
      M(I,J,K)=M2
      GO TO 44
  244 CONTINUE
      DELZ=T(I,J,K+1)+T(I,J,K-1)-2.*T(I,J,K)
      M(I,J,K)=M(I,J,K)+.5*TSI*(2.*(V(I,JP1,K)-V(I,J,K))-DELZ)
   44 CONTINUE
C
C     INSULATED FACE AT Z=0
C
      K=1
      KP1=K+1
      DO 45 I=2,N1
      IP1=I+1
      IM1=I-1
      DO 45 J=2,N1
      JP1=J+1
      JM1=J-1
      IF(LOOP.EQ.2)GO TO 145
```

```
      IF(LOOP.EQ.3)GO TO 245
      M1=.5*(T(IP1,J,K)+T(IM1,J,K)-2.*T(I,J,K))
      M2=T(I,JP1,K)+T(I,JM1,K)+2.*T(I,J,KP1)-4.*T(I,J,K)
      M(I,J,K)=T(I,J,K)+TSI*(M1+M2)
      GO TO 45
  145 CONTINUE
      DELU=U(I+1,J,K)+U(I-1,J,K)-2.*U(I,J,K)
      DELY=T(I,J+1,K)+T(I,J-1,K)-2.*T(I,J,K)
      M(I,J,K)=M(I,J,K)+.5*TSI*(DELU-DELY)
      GO TO 45
  245 CONTINUE
      DELV=V(I,J+1,K)+V(I,J-1,K)-2.*V(I,J,K)
      M(I,J,K)=M(I,J,K)+.5*TSI*(DELV-2.*(T(I,J,KP1)-T(I,J,K)))
   45 CONTINUE
C
C     AT THE FACE WHERE Y=0
C
      J=1
      JP1=J+1
      DO 46 I=2,N1
      IP1=I+1
      IM1=I-1
      DO 46 K=2,N1
      KP1=K+1
      KM1=K-1
      IF(LOOP.EQ.2)GO TO 146
      IF(LOOP.EQ.3)GO TO 246
      M1=.5*(T(IP1,J,K)+T(IM1,J,K)-2.*T(I,J,K))
      M2=2.*T(I,JP1,K)+T(I,J,KP1)+T(I,J,KM1)-4.*T(I,J,K)
      M(I,J,K)=T(I,J,K)+TSI*(M1+M2)
      GO TO 46
  146 CONTINUE
      DELU=U(I+1,J,K)+U(I-1,J,K)-2.*U(I,J,K)
      M(I,J,K)=M(I,J,K)+.5*TSI*(DELU-2.*(T(I,JP1,K)-T(I,J,K)))
      GO TO 46
  246 CONTINUE
      DELZ=T(I,J,K+1)+T(I,J,K-1)-2.*T(I,J,K)
      M(I,J,K)=M(I,J,K)+.5*TSI*(2.*(V(I,JP1,K)-V(I,J,K))-DELZ)
   46 CONTINUE
C
C     AT THE FACE WHERE X=0
C
      I=1
      IP1=I+1
      DO 47 J=2,N1
      JP1=J+1
      JM1=J-1
      DO 47 K=2,N1
      KP1=K+1
      KM1=K-1
      IF(LOOP.EQ.2)GO TO 147
      IF(LOOP.EQ.3)GO TO 247
      M1=T(IP1,J,K)+T(I,JP1,K)+T(I,J,KP1)+T(I,J,KM1)+T(I,JM1,K)
      M(I,J,K)=T(I,J,K)+TSI*(M1-5.*T(I,J,K))
```

```
      GO TO 47
  147 CONTINUE
      DELY=T(I,J+1,K)+T(I,J-1,K)-2.*T(I,J,K)
      M(I,J,K)=M(I,J,K)+.5*TSI*(2.*(U(IP1,J,K)-U(I,J,K))-DELY)
      GO TO 47
  247 CONTINUE
      DELV=V(I,J+1,K)+V(I,J-1,K)-2.*V(I,J,K)
      DELZ=T(I,J,K+1)+T(I,J,K-1)-2.*T(I,J,K)
      M(I,J,K)=M(I,J,K)+.5*TSI*(DELV-DELZ)
   47 CONTINUE
C
C     CORNER AT ORIGIN
C
      IF(LOOP.EQ.2) GO TO 151
      IF(LOOP.EQ.3)GO TO 251
      M1=T(2,1,1)+2.*T(1,2,1)+2.*T(1,1,2)-5.*T(1,1,1)
      M(1,1,1)=T(1,1,1)+TSI*M1
      GO TO 51
  151 CONTINUE
      M1=M(1,1,1)
      M2=M1+.5*TSI*(2.*(U(2,1,1)-U(1,1,1))-2.*(T(1,2,1)-T(1,1,1)))
      M(1,1,1)=M2
      GO TO 51
  251 CONTINUE
      M1=M(1,1,1)
      M2=M1+.5*TSI*(2.*(V(1,2,1)-V(1,1,1))-2.*(T(1,1,2)-T(1,1,1)))
      M(1,1,1)=M2
   51 CONTINUE
C
C
      IF(LOOP.EQ.2)GO TO 130
      IF(LOOP.EQ.3)GO TO 230
C
      DO 70 K=1,N1
      DO 70 J=1,N1
      DO 30 I=1,N1
   30 D(I)=M(I,J,K)
C
C     CALL TRI DIAGONAL MATRIX ALGORITHM
C
      CALL TRIDAG(1,N1,A,B,C,D,TEMP)
C
      DO 430 I=1,N1
  430 U(I,J,K)=TEMP(I)
C
   70 CONTINUE
      GO TO 330
  130 CONTINUE
      DO 71 K=1,N1
      DO 71 I=1,N1
      DO 72 J=1,N1
   72 D(J)=M(I,J,K)
C
      CALL TRIDAG(1,N1,A,B,C,D,TEMP)
```

```
C
      DO 472 J=1,N1
 472  V(I,J,K)=TEMP(J)
C
 71   CONTINUE
      GO TO 330
 230  CONTINUE
      DO 73 I=1,N1
      DO 73 J=1,N1
      DO 74 K=1,N1
 74   D(K)=M(I,J,K)
C
      CALL TRIDAG(1,N1,A,B,C,D,TEMP)
C
      DO 474 K=1,N1
 474  T(I,J,K)=TEMP(K)
C
 73   CONTINUE
 330  CONTINUE
 120  CONTINUE
      ITMAX=ITMAX+1
      IF(ITMAX.LT.IPR)GO TO 20
      WRITE(6,5)
 5    FORMAT(/)
      WRITE(6,91)L
 91   FORMAT(1X,'TIME STEP NUMBER=',I3/)
      TIME=DT*DFLOAT(L)
      WRITE(6,32)TIME
 32   FORMAT(5X,'ELAPSED TIME=',F10.4,'SECONDS'/)
C
C     PRINT OUT THE DIAGONAL RESULTS
C
      WRITE(6,82)(T(I,I,I),I=1,N)
 82   FORMAT(11(2X,F8.6))
      ITMAX=0
 20   CONTINUE
      RETURN
      END
C
C     SUBROUTINE TRIDAG
C
      SUBROUTINE TRIDAG(IF,L,A,B,C,D,V)
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 A(25),B(25),C(25),D(25),V(25),BETA(25),GAMMA(25)
C
C     COMPUTE INTERMEDIATE ARRAYS BETTA AND GAMMA
C
      BETA(IF)=B(IF)
      GAMMA(IF)=D(IF)/BETA(IF)
      IFP1=IF+1
      DO 1 I=IFP1,L
      BETA(I)=B(I)-A(I)*C(I-1)/BETA(I-1)
 1    GAMMA(I)=(D(I)-A(I)*GAMMA(I-1))/BETA(I)
C
```

```
C       COMPUTE FINAL SOLUTION VECTOR V
C
        V(L)=GAMMA(L)
        LAST=L-IF
        DO 2 K=1,LAST
        I=L-K
 2      V(I)=GAMMA(I)-C(I)*V(I+1)/BETA(I)
        RETURN
        END
```

```
C
C     WRITTEN BY R.F. HANDSCHUH
C
C     SOURCE.BURGER
C
C     ***** PROGRAM #12 *****
C
C     THIS PROGRAM IS FOR THE SOLUTION OF BURGER'S EQUATION
C     BY THE EXPONENTIAL FINITE DIFFERENCE METHOD.
C
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 V(100)
C
C     READ IN DATA TO BE USED IN THE SOLUTION
C
      WRITE(6,15)
 15   FORMAT(1X,'NUMBER OF NODES=N   I3'/)
      READ(9,10)N
 10   FORMAT(I3)
      WRITE(6,12)
 12   FORMAT(1X,'NUMBER OF TIME SUB INTERVALS= NS  I3')
      READ(9,13)NS
 13   FORMAT(I3)
      WRITE(6,16)
 16   FORMAT(1X,'TOTAL NUMBER OF TIME STEPS= NTOT   I3')
      READ(9,21)NTOT
 21   FORMAT(I3)
      WRITE(6,24)
 24   FORMAT(1X,'INPUT TIME / LENGTH SQUARED  F5.3')
      READ(9,25)TSI
 25   FORMAT(F5.3)
      WRITE(6,26)
 26   FORMAT(1X,'INPUT KINEMATIC VISCOSITY= F5.3')
      READ(9,27)RNU
 27   FORMAT(F5.3)
      WRITE(6,22)
 22   FORMAT(1X,'TOTAL TIME OF ONE TIME STEP= T F5.3')
      READ(9,23)T
 23   FORMAT(F5.3)
      WRITE(6,14)
 14   FORMAT(1X,'INPUT NUMBUR OF STEPS BETWEEN PRINTS=I3')
      READ(9,17)IPR
 17   FORMAT(I3)
C
C     DATA FOR INITIAL AND BOUNDRY CONDITIONS
C
      V(1)=0.
      V(N)=1.
      N1=N-1
      DO 30 I=2,N1
 30   V(I)=1.
C
C     CALL EXPONETIAL FINITE DIFFERENCE FOR BURGER'S EQUATION
```

```
C
      CALL BURG(N,NS,NTOT,TSI,V,T,RNU,IPR)
      STOP
      END
C
C     SUBROUTINE BURG
C
      SUBROUTINE BURG(N,NS,NTOT,TSI,V,T,RNU,IPR)
C
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 VT(100),V(100),M(100),P(100),THE(100)
      TS=TSI/DFLOAT(NS+1)
      DX=1.0/DFLOAT(N-1)
      N1=N-1
      NS1=NS+1
      WRITE(6,900)
  900 FORMAT(/'***** SOLUTION FOR BURGER EQUATION *****'/)
C
C     TOTAL TIME STEP LOOP
C
      DO 20 L=1,NTOT
      ITMAX=ITMAX+1
C
C     ZERO THE SUM OF DRIVE NUMBERS
C
      DO 15 I=1,N
   15 P(I)=0.
C
C     SET THE TEMPOARY FIELD VARIABLE EQUAL TO THE LAST TIME STEP VALUE
C
      DO 10 I=1,N
   10 VT(I)=V(I)
C
C     SUB TIME INTERVAL
C
      DO 30 K=1,NS1
C
C     CALCULATE THE SUB-INTERVAL DRIVE NUMBERS
C
      DO 40 I=2,N1
      IM1=I-1
      IP1=I+1
      IF(VT(I).LE.0.0)GO TO 40
      M(I)=-0.5*DX*(1.-VT(I))*(VT(IP1)-VT(IM1))/VT(I)
      M(I)=M(I)+RNU*(VT(IP1)+VT(IM1)-2.*VT(I))/VT(I)
   40 CONTINUE
C
C     CALCULATE THE SUB-INTERVAL DEPENDENT VARIABLES
C
      DO 50 I1=2,N1
      CHECK=TS*M(I1)
      IF(CHECK.LE.-50.)VT(I1)=0.0
      IF(CHECK.LE.-50.)GO TO 50
```

```
      VT(I1)=VT(I1)*DEXP(TS*M(I1))
 50   CONTINUE
C
C     SUM THE DRIVE NUMBERS
C
      DO 60 I=2,N1
 60   P(I)=P(I)+M(I)
 30   CONTINUE
C
C
C     CALCULATE THE DEPENDENT VARIABLE AT THE NEXT COMPLETE STEP
C
      DO 70 I=1,N
      CHECK=TS*P(I)
      IF(CHECK.LE.-50.)V(I)=0.0
      IF(CHECK.LE.-50.)GO TO 70
      V(I)=V(I)*DEXP(TS*P(I))
 70   CONTINUE
C
C     OUTPUT THE RESULTS
C
      IF(ITMAX.LT.IPR)GO TO 20
      ITMAX=0
      WRITE(6,5)
 5    FORMAT(/)
      WRITE(6,31)L
 31   FORMAT(1X,'TIME STEP NUMBER=',I3)
      TIME=T*DFLOAT(L)
      WRITE(6,32)TIME
 32   FORMAT(5X,'ELAPSED TIME=',F10.4,'SECONDS')
      ISTEP=(N-1)/10
      DO 110 I=1,N
 110  THE(I)=1.0-V(I)
      DO 80 J=1,ISTEP
      IS=(J-1)*11+1
      IFIN=J*10+1
      WRITE(6,81)(THE(I),I=IS,IFIN)
 81   FORMAT(1X,11(F8.6,2X))
 80   CONTINUE
 20   CONTINUE
      RETURN
      END
```

```
C
C     WRITTEN BY R.F. HANDSCHUH
C
C     SOURCE.EXBURG
C
C     ***** PROGRAM #13 *****
C
C     THIS PROGRAM IS FOR THE SOLUTION OF BURGER'S EQUATION USING AN EXPLICIT
C     TECHNIQUE.  THE RESULTS WILL BE USED TO COMPARE TO THE EXPONENTIAL
C     FINITE DIFFERENCE TECHNIQUE.
C
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 V(100)
C
C     INPUT PROGRAM DATA
C
      WRITE(6,15)
 15   FORMAT(1X,'NUMBER OF NODES=N   I3'/)
      READ(9,10)N
 10   FORMAT(I3)
      WRITE(6,16)
 16   FORMAT(1X,'TOTAL NUMBER OF TIME STEPS= NTOT   I3')
      READ(9,21)NTOT
 21   FORMAT(I3)
      WRITE(6,24)
 24   FORMAT(1X,'INPUT TIME / LENGTH SQUARED  F5.3')
      READ(9,25)TSI
 25   FORMAT(F5.3)
      WRITE(6,26)
 26   FORMAT(1X,'INPUT KINEMATIC VISCOSITY= F5.3')
      READ(9,27)RNU
 27   FORMAT(F5.3)
      WRITE(6,22)
 22   FORMAT(1X,'TOTAL TIME OF ONE TIME STEP= T F5.3')
      READ(9,23)T
 23   FORMAT(F5.3)
      WRITE(6,14)
 14   FORMAT(1X,'INPUT NUMBER OF STEPS BETWEEN PRINTS=I3')
      READ(9,17)IPR
 17   FORMAT(I3)
C
C     INITIALIZE THE BOUNDRY CONDITIONS
C
      V(1)=1.
      V(N)=0.
      N1=N-1
      DO 30 I=2,N1
 30   V(I)=0.
C
C     CALL EXPLICIT FINITE DIFFERENCE SOLUTION FOR BURGER'S EQUATION
C
      CALL BURG(N,NTOT,TSI,V,T,RNU,IPR)
      STOP
```

```
      END
C
C     SUBROUTINE BURG
C
      SUBROUTINE BURG(N,NTOT,TSI,V,T,RNU,IPR)
C
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 VT(100),V(100),THE(100)
C
C     PRINT HEADING
C
      WRITE(6,999)
 999  FORMAT(1X,'**** EXPLICIT BURGER S EQT SOLUTION ****'/)
      DX=1.0/DFLOAT(N-1)
      N1=N-1
C
C     TIME STEP LOOP
C
      DO 20 L=1,NTOT
      ITMAX=ITMAX+1
C
C     SAVE LAST TIME STEP VALUES
C
      DO 10 I=1,N
 10   VT(I)=V(I)
C
C     EVALUATE EXPLICIT FINITE DIFFERENCE EQUATION
C
      DO 40 I=2,N1
      IM1=I-1
      IP1=I+1
      V(I)=VT(I)-VT(I)*T*(VT(IP1)-VT(IM1))/(2.*DX)
      V(I)=V(I)+RNU*T*(VT(IP1)+VT(IM1)-2.*VT(I))/(DX*DX)
 40   CONTINUE
C
C     WRITE OUT THE RESULTS
C
      IF(ITMAX.LT.IPR)GO TO 20
      ITMAX=0
      WRITE(6,5)
 5    FORMAT(/)
      WRITE(6,31)L
 31   FORMAT(1X,'TIME STEP NUMBER=',I3)
      TIME=T*DFLOAT(L)
      WRITE(6,32)TIME
 32   FORMAT(5X,'ELAPSED TIME=',F10.4,'SECONDS')
      ISTEP=(N-1)/10
      DO 110 I=1,N
 110  THE(I)=V(I)
      DO 80 J=1,ISTEP
      IS=(J-1)*11+1
      IFIN=J*10+1
      WRITE(6,81)(THE(I),I=IS,IFIN)
```

```
 81   FORMAT(1X,11(F8.6,2X))
 80   CONTINUE
 20   CONTINUE
      RETURN
      END
```

```
C
C       SOURCE.NONBOU
C
C       WRITTEN BY R.F. HANDSCHUH
C
C       ***** PROGRAM #14 *****
C
C       THIS PROGRAM IS USED FOR THE SOLUTION OF THE BOUNDRY LAYER FLOW
C       OVER A FLAT PLATE. THE DIRECTION OF FLOW IS IN THE X-DIRECTION
C       WHICH IS USED AS THE MARCHING DIRECTION FOR THE EXPONENTIAL FINITE
C       DIFFERENCE ALGORITHM.  THE THERMAL AND VELOCITY BOUNDRY LAYERS
C       CAN BE EXTRACTED FROM THE TEMPERATURE AND VELOCITY FIELDS FOUND.
C
C
C
        IMPLICIT REAL*8(A-H,O-Z)
        REAL*8 U(101),T(101),V(101)
C
C       INPUT PROGRAM DATA
C
        WRITE(6,15)
   15   FORMAT(1X,'NUMBER OF NODES IN  Y DIRC=N   I3'/)
        READ(9,10)N
   10   FORMAT(I3)
        WRITE(6,12)
   12   FORMAT(1X,'NUMBER OF SUB INTERVALS= NS  I3')
        READ(9,13)NS
   13   FORMAT(I3)
        WRITE(6,16)
   16   FORMAT(1X,'TOTAL NUMBER OF X-DIR STEPS= NTOT   I3')
        READ(9,21)NTOT
   21   FORMAT(I3)
        WRITE(6,24)
   24   FORMAT(1X,'INPUT STEP LENGTH F5.3')
        READ(9,25)DX
   25   FORMAT(F5.3)
        WRITE(6,26)
   26   FORMAT(1X,'NUMBER OF STEPS BEFORE PRINTING= I3')
        READ(9,27)IPR
   27   FORMAT(I3)
        WRITE(6,110)
  110   FORMAT(1X,'INPUT KINEMATIC VISCOSITY= F6.4')
        READ(9,111) RNU
  111   FORMAT(F6.4)
        WRITE(6,101)
  101   FORMAT(1X,'INPUT THERMAL DIFFUSIVITY F6.4')
        READ(9,102)RAL
  102   FORMAT(F6.4)
        WRITE(6,103)
  103   FORMAT(1X,'INPUT YMAX F5.1')
        READ(9,104) YMAX
  104   FORMAT(F5.1)
        WRITE(6,250)N,NS,NTOT
  250   FORMAT(1X,'# OF NODES=',I3,2X,'# OF SUB-INT=',I3,2X,          -
```

```
     *'# OF TIME STEPS=',I3)
      RN=DFLOAT(N-1)
      DY=YMAX/RN
      WRITE(6,251)DX,DY
  251 FORMAT(1X,'DX=',F8.4,2X,'DY=',F8.4/)
      WRITE(6,252)RNU,RAL
  252 FORMAT(1X,'KINEMATIC VISCOSITY=',F6.4,'CM*CM/S',2X,
     *'THERMAL DIFFUSIVITY=',F6.4,'CM*CM/S')
C
C     INITIALIZE BOUNDRY CONDITIONS
C
      DO 30 J=2,N
      U(J)=1.0
      V(J)=0.0
   30 T(J)=1.0
C
      U(1)=0.0
      T(1)=0.0
C
C
      CALL NON1(N,NS,NTOT,RNU,U,V,T,IPR,DX,DY,RAL)
      STOP
      END
C
C
      SUBROUTINE NON1(N,NS,NTOT,RNU,U,V,T,IPR,DX,DY,RAL)
C
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 U(101),MU(101),V(101),T(101),MT(101)
      REAL*8 PU(101),PT(101),UT(101),TT(101),VT(101)
      REAL*8 THE(101,1000,3),UT1(101)
C
C     PRINT HEADING
C
      WRITE(6,5)
      WRITE(6,222)
  222 FORMAT(1X,'************   SOURCE.NONBOU   ************'//)
      WRITE(6,223)
  223 FORMAT(/,1X,'SOLUTION FOR BOUNDRY LAYER FLOW PAST A FLAT PLATE'/)
C
      DY2=DY*DY
      TS=DX*RAL/(DY2*DFLOAT(NS+1))
      TS1=DX*RNU/(DY2*DFLOAT(NS+1))
      DEL=DX/DFLOAT(NS+1)
      N1=N-1
      NS1=NS+1
      NSTEP=(N-1)/10
C
C     BEGIN TOTAL INTERVAL LOOP FOR L=1 TO NTOT STEPS
C
      DO 20 L=1,NTOT
C
C     ZERO THE SUM OF DRIVE NUMBERS FOR THE NEXT SET OF SUB-POSITION INTERVALS
```

```
C
      DO 15 I=1,N
      PU(I)=0.0
 15   PT(I)=0.0
C
C     SAVE LAST POSITION STEP VALUES FOR TEMPORARY VARIABLE CALCULATIONS
C     ON SUB-POSITION INTERVAL
C
      DO 10 I=1,N
      VT(I)=V(I)
      UT(I)=U(I)
 10   TT(I)=T(I)
C
C     SUB - POSITION INTERVAL
C
      DO 30 K=1,NS1
C
C     CALCULATE TEMPERATURE FIELD DRIVE NUMBER
C
      DO 41 J=2,N1
      JM1=J-1
      JP1=J+1
      MT(J)=-VT(J)*(TT(JP1)-TT(JM1))*DY/(2.*RAL*UT(J)*TT(J))
      MT(J)=MT(J)+(TT(JP1)+TT(JM1)-2.*TT(J))/(UT(J)*TT(J))
 41   CONTINUE
C
C     CALCULATE X - DIRECTION VELOCITY DRIVE NUMBER
C
      DO 141 J=2,N1
      JM1=J-1
      JP1=J+1
      MU(J)=-.5*VT(J)*DY*(UT(JP1)-UT(JM1))/(RNU*UT(J)*UT(J))
      MU(J)=MU(J)+(UT(JP1)+UT(JM1)-2.*UT(J))/(UT(J)*UT(J))
 141  CONTINUE
C
C     CALCULATE TEMPERATURE, X-DIRECTION VELOCITY, AND Y-DIRECTION VELOCITY
C     ON THE SUB-POSITION INTERVAL
C
      DO 50 I1=2,N1
 50   TT(I1)=TT(I1)*DEXP(TS*MT(I1))
C
      DO 51 I=2,N1
      UT1(I)=UT(I)
 51   UT(I)=UT(I)*DEXP(TS1*MU(I))
C
      DO 65 J=2,N1
      JM1=J-1
      VT(J)=VT(JM1)-.5*(DY/DEL)*(UT(J)-UT1(J)+UT(JM1)-UT1(JM1))
 65   CONTINUE
C
C     SUM THE DRIVE NUMBERS
C
      DO 60 J=2,N1
      PU(J)=PU(J)+MU(J)
```

```
 60   PT(J)=PT(J)+MT(J)
 30   CONTINUE
C
C     CALCULATE THE NEXT TOTAL POSITION STEP VALUES OF VELOCITIES AND TEMPERATURE
C
      DO 70 J=1,N
      UT(J)=U(J)
      U(J)=U(J)*DEXP(TS1*PU(J))
      T(J)=T(J)*DEXP(TS*PT(J))
 70   CONTINUE
C
      DO 75 J=2,N1
      JM1=J-1
 75   V(J)=V(JM1)-.5*(DY/DX)*(U(J)-UT(J)+U(JM1)-UT(JM1))
C
      ITMAX=ITMAX+1
C
C     SAVE THE VALUES FOUND IN 3-DIMENSIONAL ARRAY "THE"
C
      DO 76 J=1,N
      THE(J,L,1)=U(J)
      THE(J,L,2)=V(J)
 76   THE(J,L,3)=T(J)
      IF(ITMAX.LT.IPR)GO TO 20
C
C     WRITE OUT THE RESULTS AT THE REQUESTED INTERVAL OF POSITION
C
      WRITE(6,5)
 5    FORMAT(/)
      WRITE(6,31)L
 31   FORMAT(5X,'POSITION STEP NUMBER=',I3)
      TSTEP=DX*DFLOAT(L)
      WRITE(6,32)TSTEP
 32   FORMAT(5X,'X-POSITION=',F10.4/)
      WRITE(6,101)
 101  FORMAT(1X,'THE U VELOCITY COMPONENT')
      DO 300 KK=1,NSTEP
      IS=(KK-1)*11+1
      IFIN=KK*10+KK
      WRITE(6,82)(THE(I,L,1),I=IS,IFIN)
 300  CONTINUE
 82   FORMAT(11(2X,F8.5))
      WRITE(6,102)
 102  FORMAT(1X,'THE V VELOCITY COMPONENT')
      DO 301 KK=1,NSTEP
      IS=(KK-1)*11+1
      IFIN=KK*10+KK
      WRITE(6,82)(THE(I,L,2),I=IS,IFIN)
 301  CONTINUE
      WRITE(6,103)
 103  FORMAT(1X,'THE T FIELD VARIABLE ')
      DO 302 KK=1,NSTEP
      IS=(KK-1)*11+1
      IFIN=KK*10+KK
```

```
      WRITE(6,82)(THE(I,L,3),I=IS,IFIN)
302   CONTINUE
      ITMAX=0
20    CONTINUE
      RETURN
      END
```

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-89874 AVSCOM TR-87-C-19		

4. Title and Subtitle	5. Report Date
An Exponential Finite Difference Technique for Solving Partial Differential Equations	June 1987
	6. Performing Organization Code 505-63-51
7. Author(s) Robert F. Handschuh	8. Performing Organization Report No. E-3544
	10. Work Unit No.
9. Performing Organization Name and Address National Aeronautics and Space Administration Lewis Research Center Cleveland, Ohio 44135	11. Contract or Grant No.
	13. Type of Report and Period Covered Technical Memorandum
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, D.C. 20546	14. Sponsoring Agency Code

15. Supplementary Notes

This report was a thesis submitted in partial fulfillment of the requirements of the Master of Science Degree in Mechanical Engineering to The University of Toledo, Toledo, Ohio.

16. Abstract

An exponential finite difference algorithm, as first presented by Bhattacharya for one-dimensional unsteady state, heat conduction in Cartesian coordinates, has been extended. The finite difference algorithm developed was used to solve the diffusion equation in one-dimensional cylindrical coordinates and applied to two- and three-dimensional problems in Cartesian coordinates. The method was also used to solve nonlinear partial differential equations in one (Burger's equation) and two (Boundary Layer equations) dimensional Cartesian coordinates. Predicted results were compared to exact solutions where available, or to results obtained by other numerical methods. It was found that the exponential finite difference method produced results that were more accurate than those obtained by other numerical methods, especially during the initial transient portion of the soltion. Other applications made using the exponential finite difference technique included unsteady one-dimensional heat transfer with temperature varying thermal conductivity and the development of the temperature field in a laminar Couette flow.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Numerical methods Exponential finite difference Heat transfer	Unclassified - unlimited STAR Category 64

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No of pages	22. Price*
Unclassified	Unclassified	134	A07

NASA FORM 1626 OCT 86

*For sale by the National Technical Information Service, Springfield, Virginia 22161

END

9-87

DTIC